



## 저작자표시-비영리-동일조건변경허락 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



동일조건변경허락. 귀하가 이 저작물을 개작, 변형 또는 가공했을 경우에는, 이 저작물과 동일한 이용허락조건하에서만 배포할 수 있습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

이학박사 학위논문

A Study on Efficient Algorithms for  
the Numerical Simulation of Thin  
Film Solar Cells

박막형 태양전지 수치해석을 위한 효율적  
알고리즘에 관한 연구

2013 년 8 월

서울대학교 대학원

협동과정 계산과학전공

변 석 용

# A Study on Efficient Algorithms for the Numerical Simulation of Thin Film Solar Cells

지도 교수 신 동 우

이 논문을 이학박사 학위논문으로 제출함  
2013 년 5 월

서울대학교 대학원  
협동과정 계산과학전공  
변 석 용

변석용의 이학박사 학위논문을 인준함  
2013 년 6 월

위 원 장 정 현 교 (인)

부위원장 신 동 우 (인)

위 원 고 형 석 (인)

위 원 박 춘 재 (인)

위 원 김 임 범 (인)

# A Study on Efficient Algorithms for the Numerical Simulation of Thin Film Solar Cells

A Dissertation Submitted

by

Seok Yong Byun

to

The Interdisciplinary Program in Computational Science and Technology

in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

Graduate School

Seoul National University

August 2013

## Abstract

In this thesis, I proposed a novel intersection algorithm based absorption energy simulation methods for thin film solar cells which use a 3-D randomly textured geometry or plasma effects.

For the case of pyramidal textured thin film solar cells, Optimizing the design of the surface texture is an essential aspect of the thin film Si solar cells technology as it can maximize the light trapping efficiency of the cells. Thus, the appropriate simulation tools can provide efficient means of designing and analyzing the effects of the texture patterns on light confinement in an active medium. A ray tracing method is a powerful numerical simulation methodology for this. However, in past researches, a real object intersection method take an  $O(N^2)$  time complexity and some height map method take an  $O(N)$  time complexity. These are time consuming process and inaccurate process, so I developed a novel intersection algorithm with an  $O(\log N)$  time complexity and with keeping the accuracy. Also, an absorption energy calculation algorithm for each layer with a direct method did not exist in the past. To solve the intersection finding problem, I proposed a novel and an efficient 3-D texture intersection algorithm using a modified kd-tree traversal method in Chapter 2. Also, to solve the absorption efficiency calculation problem with ray tracing method, I proposed a new method in Chapter 3. The correctness and efficiency of the algorithms was validated by a measured data and numerical simulations.

The thickness of the thin film solar cells reach to the nanometer size. The ray tracing method is useless for the nanometer size systems except for a flat surface type. In this case, the FDTD method can be used to solve this nanometer scale problems. However, by the past researches, an auto-discretization problem and an absorption efficiency calculation problem were not solved efficiently. In this research, I proposed a robust and an efficient auto-discretization algorithm and an efficient absorption energy calculation algorithm with a continuous boundary extraction algorithm in Chapter 4. The

correctness and efficiency of the algorithms was validated by an exact solution and numerical simulations.

Through this thesis, I proposed an efficient absorption efficiency calculation algorithms for all system ranges of the thin film solar cells.

**Keywords :** Thin film solar cells, Ray tracing, Finite difference time domain, Intersection, Optical absorption efficiency

**Student Number :** 2010-20407

## **Publications**

The work in this thesis is based on the following publications :

- S.-J. Byun, S. Y. Byun, J. Lee, J. W. Kim, T. S. Lee, W. M. Kim, Y. K. Park, and K. Cho, *An optical simulation algorithm based on ray tracing technique for light absorption in thin film solar cells*, Sol. Energy Mater. Sol. Cells 95, pp.408-411, 2011.
- S. -J. Byun, S. Y. Byun, J. K. Lee, J. W. Kim, T. S. Lee, K. M. Cho. D. Sheen, S. J. Tark, D. H. Kim, W. M. Kim, *Analysis of light trapping effects in Si solar cells with a textured surface by ray tracing simulation*, Current Applied Physics, pp. 1-3, 2011.
- S. Y. Byun, S.-J. Byun, J. K. Lee, J. W. Kim, T. S. Lee, D. Sheen, K. Cho, S. J. Park, D. Kim and W. M. Kim, *An efficient ray tracing algorithm for the simulation of light trapping effects in Si solar cells with textured surfaces*. J. Nanosci. Nanotech., Vol. 12, pp. 3224-3227, 2012.
- S. -J. Byun, S. Y. Byun, T.S. Lee, W.M. Kim, K. Cho, D. Sheen, S.J. Tark, D. Kim, *High-efficiency Grid-type Si Solar Cell Structure*, J. Kor. Phys. Soc., vol. 60, pp. 2075–2078, 2012.
- S. Y. Byun, S.-J. Byun, T. S. Lee, D. Sheen, J.-H. Jeong, W. M. Kim, *Effect of oxide thin films in back contact on the optical absorption efficiency of thin crystalline Si solar cells*, Current Applied Physics (2013), <http://dx.doi.org/10.1016/j.cap.2013.01.045>.

# Table of Contents

Abstract .....	ii
Publications.....	iv
Table of Contents .....	v
List of Figures .....	ix
List of Tables.....	xiii
List of Algorithms .....	xiv
Symbols.....	xv
Abbreviations .....	xviii
1. Introduction.....	1
1.1 Motivation.....	1
1.2 Thin Film Solar cells .....	2
1.2.1 Reduction of Front Surface Reflectance .....	4
1.2.2 Enhancement of Back Surface Reflectance .....	4
1.2.3 Efficient Light Trapping.....	5
1.3 Ray Tracing .....	5
1.3.1 Finding Intersection .....	8
1.3.1.1 Primitive Object Case.....	9
1.3.1.2 CSG Object Case.....	11
1.3.2 Acceleration Scheme.....	12
1.4 Finite Difference Time Domain (FDTD) .....	18
1.4.1 Discretization of the System Domain .....	21
1.4.2 Dispersive Materials .....	21
1.4.2.1 Lorentz Model .....	22
1.4.2.2 Drude Model .....	24
1.4.2.3 Drude-Lorentz Model.....	24
1.4.3 Boundary Condition.....	25
1.4.3.1 Absorbing Boundary Condition (ABC).....	25
1.4.3.2 Periodic Boundary Condition (PBC).....	27
1.5 Scope and Objectives .....	28



1.6 Achievements .....	28
2. Slab-Outline Algorithm for Fast Intersection Finding .....	29
2.1 Overview .....	29
2.2 Algorithm .....	30
2.2.1 Non-overlapped texture case .....	31
2.2.2 Overlapped pyramidal texture case .....	43
2.3 Numerical Results : Validation .....	49
2.3.1 Examine of Backward Ray Tracing Results .....	49
2.3.2 Comparison of Experimental Results .....	51
2.3.3 Error Analysis .....	53
2.3.4 Time Complexity .....	54
2.4 Numerical Analysis : Applications .....	56
2.4.1 Simulation .....	56
2.4.2 Results and discussion .....	57
2.5 Conclusion .....	60
3. Simulation with Ray Tracing Method .....	61
3.1 Overview .....	61
3.2 Algorithm .....	62
3.3 Validation .....	64
3.3.1 Case I - coherent system .....	64
3.3.2 Case II - incoherent system .....	65
3.3.3 Case III - coherent + incoherent complex system .....	66
3.4 Numerical Analysis : Applications .....	68
3.4.1 High-efficiency Grid-type Si Solar Cell Structure .....	68
3.4.1.1 Overview .....	68
3.4.1.2 Simulation model .....	68
3.4.1.3 Results and Discussion .....	70
3.4.2 Effect of oxide thin films in back contact on the optical absorption efficiency of thin crystalline Si solar cells .....	73
3.4.2.1 Overview .....	73
3.4.2.2 Simulation model .....	74

3.4.2.3 Results and Discussion .....	76
3.5 Conclusion.....	81
4. Simulation with FDTD Method .....	82
4.1 Overview .....	82
4.2 Auto-Discretization of the System Domain .....	82
4.2.1 Algorithm.....	82
4.2.2 Results of Auto-Discretization .....	88
4.3 Implementation of Lorentz Model with ADE .....	90
4.4 EffectiveMaterial Function.....	94
4.4.1 Round-Off Algorithm.....	95
4.4.2 Dispersive Conformal FDTD (D-CFDTD) Algorithm.....	96
4.4.3 Validation .....	101
4.4.4 Numerical Analysis .....	105
4.5 Simulation of Absorption Energy .....	108
4.5.1 Algorithm.....	110
4.5.1.1 Extract Object's Continuous Boundary .....	111
4.5.1.2 Memory allocation and index mapping for the boundary cells .....	117
4.5.1.3 Calculation of the absorption energy.....	118
4.5.2 Numerical Analysis .....	120
4.5.2.1 Flat system.....	121
4.5.2.2 Non-Flat system. ....	124
4.6 Conclusion.....	126
5. Conclusion .....	127
5.1 Summary .....	127
5.2 Evaluation .....	127
5.3 Future Work.....	128
References .....	129
Appendix I .....	141
국문초록.....	143
감사의 글 .....	145



## List of Figures

Figure 1. Basic structure of a silicon based solar cells.....	2
Figure 2. Spectra chart of absorption length for Si at 300K [2]. .....	3
Figure 3. Simple optical system and ray tracing schematic .....	6
Figure 4. Ray directions on the material boundary [1].....	6
Figure 5. Ray structure and a simple example of the collision detection between a ray and a primitive Box object. ....	8
Figure 6. Four types of CSG operations with Box and Sphere. ....	11
Figure 7. Objects distribution in 2-D and related kd-tree structure.....	13
Figure 8. Scheme of conventional kd-tree traversal with CSG Object. ....	16
Figure 9. Schematic sketch of a thin-film microcrystalline silicon solar cell (a) on a smooth substrate and (b) on a randomly textured substrate [39]....	17
Figure 10. Example of the kd-tree construction for textured silicon solar cells. .....	17
Figure 11. Positions of various field components. The E-fields are in the middle of the edges and the H-fields are in the center of the faces [14]	20
Figure 12. Periodic structures in one-dimensional (1-D), two-dimensional (2- D), and three-dimensional (3-D) configurations [1]. ....	27
Figure 13. Two-dimensional electromagnetic screen (left) and a top view of the structure [13]. ....	27
Figure 14. Cross section view of 3-D textured solar cell. ....	31
Figure 15. Schema for non-overlapped texture of silicon solar cells. (a) is normal pyramidal shape, and (b) is inverse pyramidal shape. ....	32
Figure 16. Example of CSG tree and operation sequences. ....	33
Figure 17. Modified kd-tree with the Slab-Outline algorithm. ....	34
Figure 18. Three cases of intersection between a ray and a Slab-Outline CSG union object. ....	39
Figure 19. Three cases of intersection between a ray and a Slab-Outline CSG difference object. ....	42

Figure 20. Schema for overlapped texture of silicon solar cells. (a) is normal pyramidal shape, and (b) is inverse pyramidal shape.....	43
Figure 21. SEM images of pyramidal texture of silicon solar cells. ....	44
Figure 22. Result of the Algorithm 8. ....	49
Figure 23. Geometry of backward ray tracing system. (a) is textured silicon, (b) is box light, (c) is a chamber, and (d) is top surface geometry of the (a). ....	50
Figure 24. Simulation results of the backward ray tracing.....	50
Figure 25. (a) Surface morphology of a textured Si sample and (b) a 3-D rendering image of the geometric information for the simulation [12]..	52
Figure 26. Comparison of the measured and simulated reflectance spectra [12]. ....	52
Figure 27. Relative errors according to the increased number of rays.....	54
Figure 28. Performance comparison of Slab-Outline against CSG algorithm [12]. ....	56
Figure 29. Schematics of the cross-section (a) and plan view (b) for the simulation geometry, and (c) optical simulation structure for the reflectance and absorption [44]. ....	57
Figure 30. Simulated reflectance spectra of textured surface with various spaces (a) and pyramid sizes(b). (c) and (d) are the average absorptance corresponding to (a) and (d), respectively [44]. ....	58
Figure 31. SEM micrographs of Si surfaces etched for (a) 10, (b) 20, (c) 30 and (d) 60 min. (e) The measured reflectance spectra of the etched Si surfaces [44]. ....	59
Figure 32. (a)Schematics of tested solar cell structure, (b) spectra of absorption ratios determined from arithmetic ratio calculation (symbols) and non-sequential ray tracing technique (lines) [29]. ....	64
Figure 33. Calculated absolute absorption energy spectra for coherent (a) and incoherent (b) conditions [29]. ....	65
Figure 34. (a) Schematics and perspective view of simulation system, (b) Example of a ray path inside simulation system, and (c) Calculated	

absolute absorption energy spectra by combining coherent and incoherent ray conditions [29].....	67
Figure 35. Schematics of the six different cell structures examined in simulation [85]. ....	68
Figure 36. Ray path in Si solar cell with a pyramidal surface texture [86]. ...	69
Figure 37. Ray path in a grid-type cell structure with reflectors [85]. ....	70
Figure 38. Calculated absorptance spectra of the cell structures shown in Figure 35 [85]. ....	71
Figure 39. Comparison of the absorptance ratios of structures (b) and (f) as functions of the incident angle of the incoming light [85]. ....	72
Figure 40. (a) Example of ray paths inside the simulation system, and (b) the schematics of simulated structure [99]. ....	75
Figure 41. Optical constants of oxide films used in the simulation [99]. ....	76
Figure 42. Simulated reflection and absorption spectra for c-Si/Al structure with specular and Lambertian surfaces and those for c-Si/ZnO1/Al and c-Si/ZnO1/Ag structure with Lambertian surfaces [99]. ....	77
Figure 43. Variation of the calculated current densities (a) in c-Si by absorption, (b) reflection, and (c) in metal and oxide layers as a function of oxide layer thickness. The closed symbols shown in (a) designate the current densities absorbed in c-Si for structures with Ag as metal layer [99]. ....	79
Figure 44. Simple schema of the Algorithm 10 in 2-D case. ....	86
Figure 45. Mesh truncation of a dielectric object [47]. ....	88
Figure 46. Auto-Discretization of cylinder object in 2-D. ....	89
Figure 47. Auto-Discretization of pyramidal object in 2-D. ....	89
Figure 48. Auto-Discretization of sphere object in 3-D. ....	90
Figure 49. Auto-Discretization of Moth-eye object in 3-D. ....	90
Figure 50. Schematic illustration of the crystalline silicon sphere in free space. (a) is 2D projection view. (b) is 3D view. The green plane is detector to monitor field values. The outer box is system boundary and the inner box is far-field detector. ....	102

Figure 51. (a) Real and (b) Imaginary permittivity fitting values of the c-Si are compared with experimental data. ....	103
Figure 52. Comparison of Round-Off and D-CFDTD with Mie theory. (a) is an angular scattering cross section(SCS) values. (b) is absolute errors. ....	104
Figure 53. Root mean square errors of the three kinds of algorithms. ....	105
Figure 54. Schematic illustration of the a-Si sphere immersed c-Si cube in free space. (a) is 2D projection view. (b) is 3D view.....	106
Figure 55. (a) Real and (b) Imaginary permittivity fitting values of the a-Si are compared with experimental data. ....	107
Figure 56. Comparison of Round-Off and D-CFDTD with more fine mesh D-CFDTD. (a) is an angular scattering cross section(SCS) values. (b) is absolute errors. ....	108
Figure 57. Schematic of the discretization of immersed two objects in the FDTD system. ....	115
Figure 58. Illustration of the results of the algorithm 14 with the FDTD system of Fig. 55. (a) is 20nm cell size, (b) is 10nm cell size, and (c) is 5nm cell size.....	117
Figure 59. Schematics of the extraction of the boundary cell indices.....	118
Figure 60. Layout for the 2-D FDTD model of the flat thin film amorphous silicon solar cell.....	121
Figure 61. (a) Comparison results of the FDTD and Ray-tracing, (b) Absolute error values.....	123
Figure 62. (a) Results of the FDTD, (b) Image from [113], (c) Image from [110]. ....	123
Figure 63. Layout for the 2-D FDTD model of the non-flat thin film amorphous silicon solar cell.....	124
Figure 64. (a) Spectral comparison of absorption energy ratio, (b) Comparison of total absorption energy ratio. ....	125

# List of Tables

Table 1. Error Analysis of Slab-Outline Algorithm..... 53

Table 2. List of relative Si volume, absorptance (relative absorption energy)  
and absorptance normalized by the Si volume [85]. ..... 71

Table 3. Lorentz parameters of c-Si. .... 103

Table 4. Lorentz parameters of a-Si, (a) is four pole pairs model and (b) is five  
pole pairs model. .... 106

Table 5. (a) Lorentz parameters for ITO and (b) for AZO. .... 122



## List of Algorithms

Algorithm 1. Ray-Box Intersection .....	9
Algorithm 2. Recursive KD-tree construction [5] .....	14
Algorithm 3. Traversal KD-tree .....	15
Algorithm 4. Construct CSG object .....	35
Algorithm 5. Traversal Sub KD-tree .....	36
Algorithm 6. Traversal of Slab-Outline CSG Union object .....	37
Algorithm 7. Traversal of Slab-Outline CSG Difference object .....	40
Algorithm 8. Traversal Sub KD-tree for overlapped texture .....	44
Algorithm 9. Traversal of Slab-Outline CSG Union object .....	46
Algorithm 10. Traversal of Slab-Outline CSG Difference object .....	47
Algorithm 11. Discretize of FDTD system .....	83
Algorithm 12. Update of ADE Parameter .....	93
Algorithm 13. Update of Polarization Current .....	93
Algorithm 14. Summation of Gamma value .....	94
Algorithm 15. Round-Off method for EffectiveMaterial .....	95
Algorithm 16. Update of ADE Parameter for D-CFDTD .....	98
Algorithm 17. Update of Polarization Current for D-CFDTD .....	99
Algorithm 18. Summation of Gamma Values .....	100
Algorithm 19. Dispersive Conformal FDTD method for EffectiveMaterial .....	101
Algorithm 20. Modified Discretize method for FDTD System .....	111
Algorithm 21. Make Continuous Boundary .....	115
Algorithm 22. Calculate Absorption Energy .....	119
Algorithm 23. Improved Calculation of Absorption Energy .....	119

## Symbols

- $h$  : Planck's constant
- $\nu$  : Frequency
- $c$  : Speed of light
- $\lambda$  : Wavelength
- $n$  : Index of Refraction
- $R$  : Reflectance
- $T$  : Transmittance
- $A$  : Absorptance
- $A_{\text{flux}}$  : Flux Absorbance
- $\Phi_0(\lambda)$  : AM-1.5 Spectrum
- $H$  : Magnetic Field (amperes / meter)
- $E$  : Electric Field (volts / meter)
- $D$  : Electric Flux Density (coulombs / meter<sup>2</sup>)
- $B$  : Magnetic Flux Density (webers / meter<sup>2</sup>)
- $J$  : Electric Current Density (amperes / meter<sup>2</sup>)
- $M$  : Magnetic Current Density (volts / meter<sup>2</sup>)
- $\rho$  : Charge Density
- $\epsilon$  : Electric Permittivity (farads / meter)
- $\epsilon_0$  : Electric Permittivity in vacuum (farads / meter)
- $\epsilon''$  : Imaginary part of complex permittivity
- $\mu$  : Magnetic Permeability (henrys / meter)
- $\mu_0$  : Magnetic Permeability in vacuum (henrys / meter)
- $\mu''$  : Imaginary part of complex permeability
- $\sigma$  : Electric Conductivity (siemens / meter)
- $\sigma^*$  : Magnetic Conductivity (ohms / meter)
- $S$  : Courant number
- $\chi$  : Susceptibility
- $\omega$  : Angular frequency

- $\omega_p$ : Plasma frequency of the Lorentz pole pair
- $\delta_p$ : Damping coefficient of the Lorentz pole pair
- $\Delta\epsilon_p$ : Change in relative permittivity due to the Lorentz pole pair
- $\epsilon_\infty$ : Permittivity when wavelength goes to infinity
- $\epsilon^{\text{eff}}$ : Effective Permittivity
- $J_p$ : Polarization Current associated with the Lorentz pole pair
- $\Omega_p$ : Plasma Frequency of material
- $N_r$ : Number of Rays
- $N_i$ : Average Number of Intersections
- $N_o$ : Number of Objects
- $\alpha$ : Absorption Coefficient
- P: Power density
- Si : Silicon
- c-Si : Crystalline Silicon
- Si : Amorphous Silicon
- Al : Aluminium
- Ag : Silver
- Au : Gold
- SiO : Silicon Monoxide
- SiO<sub>2</sub> : Silicon Oxide
- TiO<sub>2</sub> : Titanium Dioxide
- Si<sub>3</sub>N<sub>4</sub> : Silicon Nitride
- Ta<sub>2</sub>O<sub>5</sub> : Tantalum Pentoxide
- Al<sub>2</sub>O<sub>3</sub> : Aluminium Oxide
- AZO : Aluminium Zinc Oxide
- CdS : Cadmium Sulfide
- CIGS : Copper Indium Gallium Selenide
- Mo : Molybdenum
- TCO : Transparent Conducting Oxide
- ZnO : Zinc Oxide

- Sn : Stannum
- $\text{In}_2\text{O}_3$  : Indium Oxide
- Ga : Gallium
- GZO : Gallium doped Zinc Oxide
- ITO : Indium Tin Oxide

## Abbreviations

- FDTD : Finite Difference Time Domain
- FEM : Finite Element Method
- CSG : Constructive Solid Geometry
- BSP : Binary Space Partitioning
- KD-Tree : K-Dimensional Tree
- BVH : Bounding Volume Hierarchy
- BIH : Bounding Interval Hierarchy
- SAH : Surface Area Heuristic
- PLRC : Piecewise Linear Recursive Convolution
- ADE : Auxiliary Differential Equation
- ABC : Absorbing Boundary Condition
- PBC : Periodic Boundary Condition
- PML : Perfectly Matched Layer
- PBG : Photonic Band Gap
- AFM : Atomic Force Microscopy
- SEM : Scanning Electron Microscope
- LSM : Least Square Method
- KOH/IPA : Potassium Hydroxide Isopropyl Alcohol
- BSDF : Bidirectional Scattering Distribution Function
- OAE : Optical Absorption Efficiency
- TIR : Total Internal Reflection
- D-CFDTD : Dispersive Conformal FDTD
- LM : Levenberg Marquardt
- RMSE : Root Mean Square Error
- DFT : Discrete Fourier Transform
- TF/SF : Total-Field / Scattered-Field
- TE : Transverse Electric
- TM : Transverse Magnetic

# 1 Introduction

## 1.1 Motivation

Many researchers tried to enhance energy conversion efficiency of solar cells in the past decades. In virtue of these efforts, the energy conversion efficiency of the solar cells is more enhanced than the earlier. However, in spite of the enhanced energy conversion efficiency, the conventional silicon solar cells have inferior economic feasibility than the other fossil energies. This poor economic feasibility is originated from the relatively expensive silicon price, and the needed amount of silicon to make the conventional silicon based solar cells.

The thin film silicon solar cells were proposed in the recent decades to solve this inferior economic feasibility problem. This new type solar cells have thinner thickness silicon layer than the conventional silicon solar cells, and this property makes possible to reduce the used silicon amount. However, flat or bare type thin film solar cells had an inferior energy conversion efficiency than the conventional solar cells. Because, the decreased thickness caused the decreased optical path length, so the entered solar energy cannot fully absorbed inside the active layer of the solar cells. Thus, to increase the optical path length, various methodologies proposed by many researchers.

Among them, a pyramidal texturing is the most generally used method with the micrometer sized thin film solar cells. Optimizing the design of the surface texture is an essential aspect of the thin film Si solar cells technology as it can increase the optical path length through maximizing the light trapping efficiency of the cells. Proper simulation tools can provide efficient means of designing and analyzing the effects of the texture patterns on light confinement in an active medium [12].

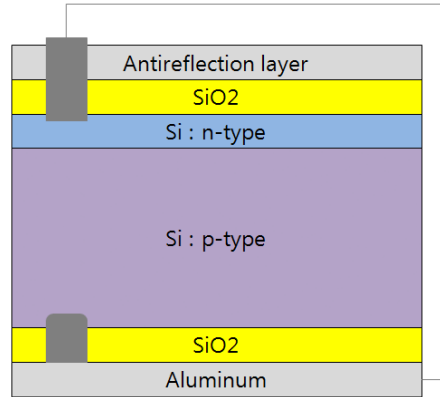
However, if the dimension of the thin film solar cells reaches the nanometer ranges, this texturing technology cannot be used. For this nanometer scale thin film solar cells, many researchers are studying various methodologies(*e.g.*, Plasmon effect, grating effect ...) to enhance the energy conversion

efficiency of the solar cells. Also, in this case, proper simulation tools play an important role in reducing the development time.

From these needs, I started my study and achieved several algorithms for the ray tracing and the finite difference time domain method that are explained in Chapter 2-4. In the following sections, I briefly introduced the background knowledge for my studies.

## 1.2 Thin Film Solar cells

A solar cell (Photovoltaic Cell) is a generator which creates electric energy from sunlight. Inside the solar cells, photons of sunlight are converted to electric energies. This conversion process occurs in an active layer within the solar cells, and in many cases, the active layer is made of crystalline or amorphous silicon materials. Figure 1 shows the basic structure of silicon based solar cells.



**Figure 1.** Basic structure of a silicon based solar cells.

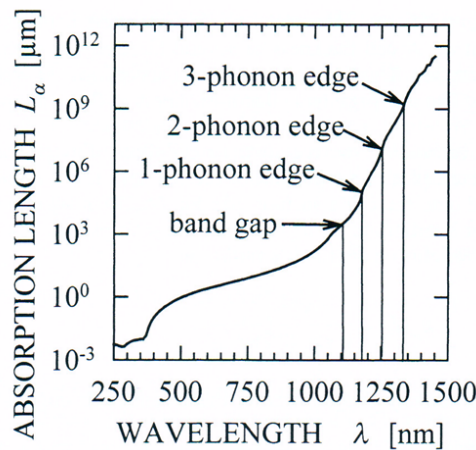
Sunlight is consisted of photons that have a broadband spectra. The photon energy is proportional to its frequency. The energy is calculated like this

$$E = h\nu = \frac{hc}{\lambda}, \quad (1.1)$$

where  $h$  is Planck's constant,  $\nu$  is the frequency of the photon,  $c$  is the speed of light and  $\lambda$  is the wavelength of the photon [3]. This photon energy is converted to electric energy when it enters the solar cells, and then it is absorbed

in the active layer. These absorbed photon energies excite electrons of the silicon material within the active layer and these electrons' movement causes of electric energies. The excited electron is called a free electron carrier. Some portions of the free electron carriers recombine holes before they reach a cathode. These recombined electron carriers cannot contribute to generate electric energy. Therefore, the energy conversion efficiency is a proportion to an absorption rate of photon energy within the active layer of the solar cells and is inverse proportion to the recombination rate of the free electron carriers.

Inside the silicon material, sunlight has different optical absorption lengths according to the wavelength. The optical absorption length is total travelling paths of the photon within the thin film solar cells until it lost its entire energy. Many conventional solar cells are composed of more than 300  $\mu\text{m}$  thick silicon wafer. Because near the 1000nm wavelengths have about 160  $\mu\text{m}$  optical absorption lengths, and also photons, which have near the 1000nm wavelengths, have the most conversion efficiency [3]. However, previously mentioned, the thick silicon wafer increases manufacturing costs of the solar cells.



**Figure 2.** Spectra chart of absorption length for Si at 300K [2].

The thin film silicon solar cells use a thin silicon film which has from nanometers to micrometers thickness. This decreased silicon thickness contributed to decrease the fabrication costs of the solar cells. However, this causes inferior conversion efficiency than the silicon wafer solar cells(*i.e.*,



conventional solar cells). This inferior energy conversion efficiency comes from a shorted optical absorption length. Like Figure 2 depicts, both visible and infrared lights of sunlight cannot be sufficiently absorbed in the thin silicon film. Therefore, the thin film solar cells need to increase the efficient optical absorption length and also need to increase the total energy conversion efficiency. The general methods of increasing the energy conversion efficiency are (i) a reduction of a front surface reflectance of the cell, (ii) an enhancement of the back surface reflectance of the cell, and (iii) an efficient light trapping that increases the optical absorption length of the photon inside the active layer [2].

### **1.2.1 Reduction of Front Surface Reflectance**

This technique is used to increase an amount of sunlight penetrated into the active layer of the thin film solar cells, and also this is commonly used in the conventional solar cells. The applied methods to obtain this purpose are optically thin layer antireflection coating, optically thick layer antireflection coating, and surface texturing [2].

The optically thin layer antireflection coating generally coats SiO, SiO<sub>2</sub>, TiO<sub>2</sub>, Si<sub>3</sub>N<sub>4</sub>, Ta<sub>2</sub>O<sub>5</sub>, or/and Al<sub>2</sub>O<sub>3</sub> on the silicon layer with sub-wavelength thickness [119-122]. The optically thick layer antireflection coating use a smaller refractive index than that of silicon [123]. The glass cover of the solar cells is an example of this. The surface texturing is frequently used reflection control method. In practice, these three techniques are used with mixed form [124, 2].

### **1.2.2 Enhancement of Back Surface Reflectance**

This technique generally uses metallic back reflector to increase the reflection ratio of out-going light through the back surface of the solar cells. If normally incident sunlight travels the active layer without any back reflector, short wavelength lights are almost absorbed inside the active layer, but long wavelength lights reach the back surface of the silicon layer and almost of them pass the surface according to the Si/Air boundary condition.

The back reflector prevent the transmitted lights and make the optical absorption length doubly with normally incident long wavelength lights. However, if metallic back reflector is used directly, then the absorption of metal reflector itself is not negligible. Therefore, the back reflector commonly is used with an almost lossless dielectric thin layer like  $\text{SiO}_2$ . This effect was researched in [99]. Also, the back reflector is commonly combined with a textured front surface to maximize its effect (see Figure 9(b)).

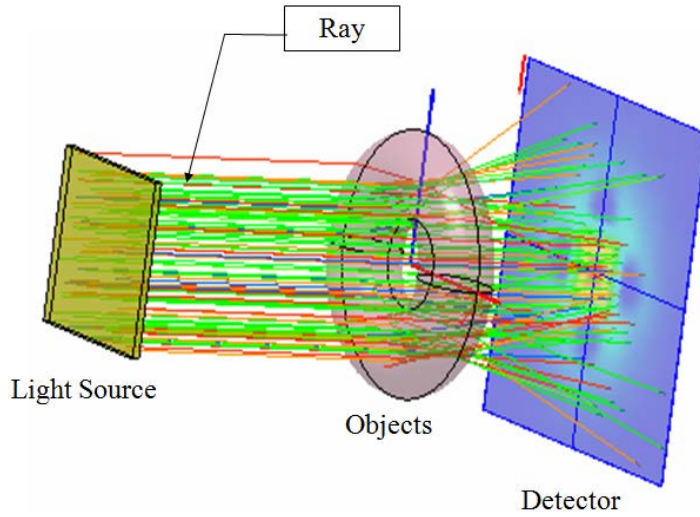
### **1.2.3 Efficient Light Trapping**

This is not a standalone method but a combined method with the front surface technique, back surface technique, plasma technique, grating (diffraction) technique, and etc. Therefore, to obtain this purpose, thin film solar cell researchers need to test a complicated solar cell system under various conditions. Thus, the researchers analyze new thin film solar cell structures and materials, which are expected to increase the energy conversion efficiency, with the above approaches. If these researches are carried out with only experiments, they are time consuming and expensive activities. Therefore, efficient numerical methods are needed, and in these days the importance is increasing.

## **1.3 Ray Tracing**

A ray tracing is a geometric optics and Monte-Carlo based simulation method. At the end of the nineteenth century, this method was used to design simple lens systems with a calculation by hands. After the emergence of computer systems, this method has been used to analyze more complex optical systems and to generate photorealistic images.

The ray tracing method uses discrete number of rays which are idealized light beams. Like Figure 3 shows, one ray starts from the light source and travels inside the optical system under the law of geometric optics. The direction and the energy of the ray is changed when the ray collides with objects inside the optical system. The ray finishes its life time when it meets detectors, or cannot meet any objects, or reaches a minimum energy limit.



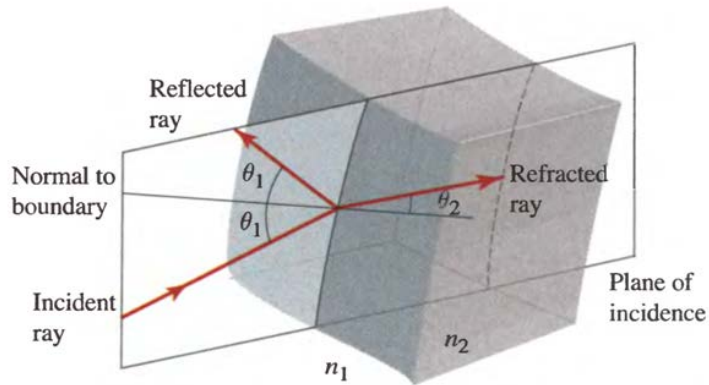
**Figure 3.** Simple optical system and ray tracing schematic

Main governing equations of the ray tracing method are Snell's law and Fresnel equations.

Snell's law is

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2} = \frac{n_2}{n_1}, \quad (1.2)$$

where  $\theta_1$  is an incident and a reflected angles of ray,  $\theta_2$  is a refracted angle of ray,  $c_1$  is a speed of incident ray,  $c_2$  is a speed of refracted ray,  $n_1$  is an index of refraction of the first material, and  $n_2$  is an index of refraction of the second material respectively [1] like as Figure 4.



**Figure 4.** Ray directions on the material boundary [1]

The direction of the ray can be determined by Snell's law, when it intersect with an object. The energy of the ray can be calculated with Fresnel equations. Fresnel equations are

$$R_s = \left| \frac{n_1 \cos \theta_1 - n_2 \cos \theta_2}{n_1 \cos \theta_1 + n_2 \cos \theta_2} \right|^2, \quad (1.3)$$

$$R_p = \left| \frac{n_1 \cos \theta_2 - n_2 \cos \theta_1}{n_1 \cos \theta_2 + n_2 \cos \theta_1} \right|^2, \quad (1.4)$$

$$R = \frac{R_s + R_p}{2}, \quad (1.5)$$

$$T = 1 - R, \quad (1.6)$$

where  $R_s$  is the reflectance of a s-polarized light,  $R_p$  is the reflectance of a p-polarized light,  $R$  is the reflectance of unpolarized light, and  $T$  is the transmittance of unpolarized light.

Except for these governing equations, the most crucial thing in the ray tracing method is a finding intersection points between the ray and objects. Because this, so called a collision detection process, is the most time consuming process in the ray tracing method. Especially, if the optical system has an enormous number of objects or some objects of the system have an enormous number of embedded particles on their surface, then the simulation time will be dramatically increased. This collision detection process was depicted in Section 1.3.1.

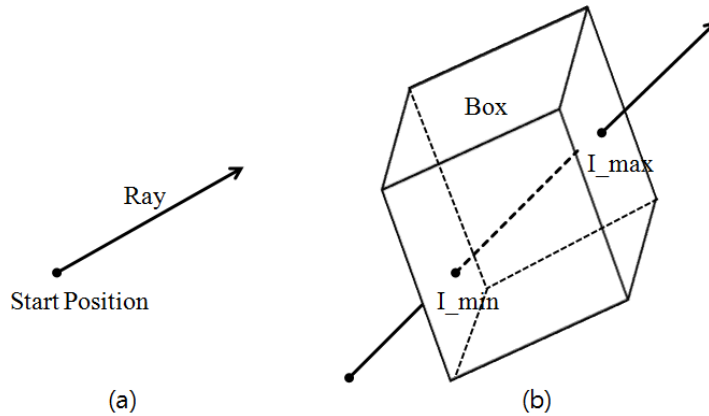
For the last several decades, many researchers developed efficient collision detection algorithms and structures [5 - 11] that are called an acceleration algorithm. A kd-tree is also one of the most famous acceleration algorithm which is used to find an intersection point between a ray and an enormous number of objects. With this algorithm, the ray tracing speed increased when the optical system has a massive number of objects or mesh structures. This acceleration algorithms were introduced in Section 1.3.2. However, this algorithm cannot be used directly for the system that has an enormous number of embedded particles on a surface of an object. These particles make bumps, holes, or immersed shapes with the object.

Unfortunately, in many cases, the active layer of the thin film solar cells has a 3-D textured structure, that is composed of pyramidal shape bumps(or

holes) with a substrate, to enhance an optical path length. To solve these kinds of problems, I developed a novel Slab-Outline algorithm to find an intersection point between a ray and a thin film solar cell with asymptotically  $O(N_r \times N_i \times \log 2N_o)$  time complexity in average cases [12]. More details are provided in Chapter 2.

### 1.3.1 Finding Intersection

The finding intersection of a geometric object with a ray is the most important procedure in ray tracing method. The thin film solar cells and other optical systems are composed of light sources, objects, and detectors (for the backward case this is a screen) like Figure 3. The ray tracing method excites rays from the light sources, and subsequently collision detection process tries to find intersection points between a ray and all objects of the system. In the ray tracing method, a ray is represented as a line equation which has a starting position and a direction vector like Figure 5(a), and objects are represented as primitive object (*e.g.*, Cube, sphere, cylinder, etc.). If an optical system needs a more complex object, this can be represented as a constructive solid geometry (CSG) [36, 37], as nurbs, or as mesh.



**Figure 5.** Ray structure and a simple example of the collision detection between a ray and a primitive Box object.

where  $I_{min}$  indicates near intersection point and  $I_{max}$  indicates a far intersection point.

#### 1.3.1.1 Primitive Object Case

The collision detection process evaluates a line and primitive object equations, whether or not these two geometry components meet like Figure 5(b). If there are intersection points, the process chooses the nearest point as a suitable intersection point. In Figure 5(b) case, the collision detection process chooses the  $I_{min}$  point as the intersection point. This ray and cube(or box) intersection algorithm is sampled at Algorithm 1 [118]. The ray tracing method has several intersection algorithms for each primitive objects like Algorithm 1, and the collision detection process use these intersection algorithms to find the valid nearest intersection point for all the objects inside the optical system.

---

**Algorithm 1.** Ray-Box Intersection

---

**function** Intersect\_Box(objects Box, Ray r, float  $I_{min}[3]$ , float  $I_{max}[3]$ )

**return** boolean

**if** Box is transformed **then**

        inverse transform Box

**end if**

$tmin = 0.0$ ,  $tmax = \text{huge value}$

    // +X, -X surface check

**if** Ray::Direction[X] < 0.0 **then**

$t = (\text{Box}::min\_Corner[X] - \text{Ray}::Origin[X]) / \text{Ray}::Direction[X]$

**if**  $t < tmin$  **then**

**return** false

**end if**

**if**  $t \leq tmax$  **then**

$tmax = t$

**end if**

$t = (\text{Box}::max\_Corner[X] - \text{Ray}::Origin[X]) / \text{Ray}::Direction[X]$

**if**  $t \geq tmin$  **then**

**if**  $t > tmax$  **then**

**return** false

**end if**

```

        tmin = t
    end if
else if Ray::Direction[X] > 0.0 then
    t = (Box::max_Corner[X] - Ray::Origin[X]) / Ray::Direction[X]
    if t < tmin then
        return false
    end if
    if t <= tmax then
        tmax = t
    end if
    t = (Box::min_Corner[X] - Ray::Origin[X]) / Ray::Direction[X]
    if t >= tmin then
        if t > tmax then
            return false
        end if
        tmin = t
    end if
else
    if Ray::Origin[X] < Box::min_Corner[X] or Ray::Origin[X]
        > Box::max_Corner[X] then
        return false
    end if
end if
// +Y, -Y, +Z, and -Z surface checks are similar.
I_min = Ray::Origin + tmin * Ray::Direction
I_max = Ray::Origin + tmax*Ray::Direction
return true
end function

```

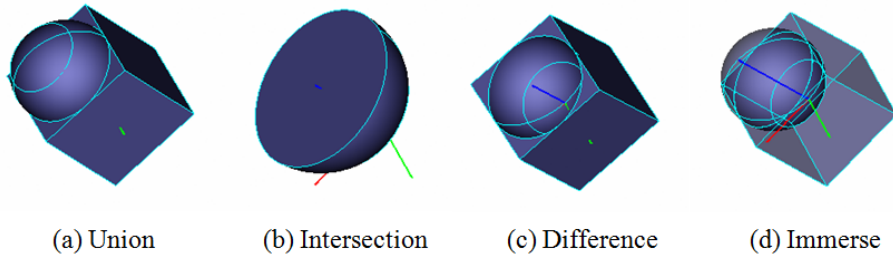
---

This evaluation process is a very time-consuming process without any kinds of an acceleration algorithm because it has to solve geometric algebra equations with all objects in the system at every steps. Thus, an efficient ac-

celeration scheme is need to reduce tracing time. This was introduced in Section 1.3.2.

#### 1.3.1.2 CSG Object Case

The CSG object is composed of the primitive objects with union, difference, intersection, and immerse boolean set operators like Figure 6. If an optical system has the CSG type object, then the collision detection process evaluate a line equation with equations of the primitive element objects included in the CSG object. This means that the collision detection algorithm has to find intersection points of the ray and all elements of the CSG object, and subsequently select a valid intersection point according to the CSG operation types. The CSG operation check means that all the founded intersection points are checked which point is located inside the other objects, and the outside points are selected as valid points.



**Figure 6.** Four types of CSG operations with Box and Sphere.

This process is more time consuming operation than the primitive case because in this case intersection check include the inside check routine for all element objects. Moreover, in this case, the conventional acceleration schemes cannot be used because all the conventional acceleration schemes assume that all objects in the optical system are not included in boolean operations each other. Therefore, if an optical system has a CSG object, it has to be treated a single object in the conventional acceleration scheme. This means that when the ray tracing method is used to analyze a textured Si solar cell, the conventional acceleration scheme is useless, so the ray tracing time will increase. To solve this problem, I developed an improved kd-tree traversal

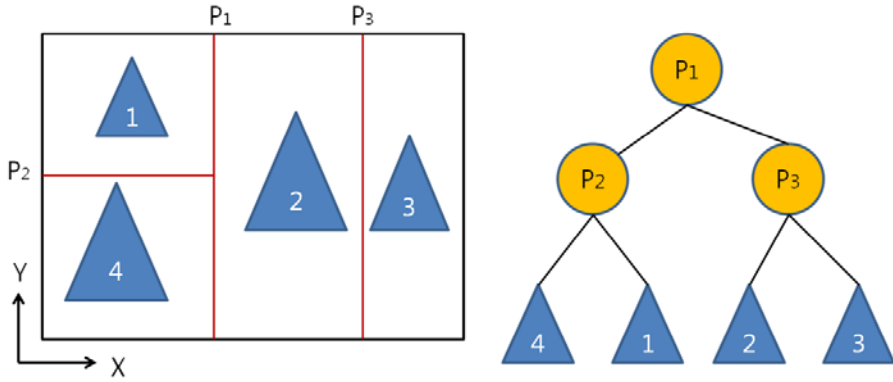


algorithm, which is the robust and efficient acceleration scheme. More detailed description showed in Chapter 2.

### **1.3.2 Acceleration Scheme**

Uniform Grid [40], Bounding Volume Hierarchy (BVH) [8-10, 41], Bounding Interval Hierarchy (BIH) [38] and K-Dimensional Tree (kd-tree) [5-7,28] are generally used acceleration schemes. These schemes commonly divide the n-dimensional system spaces to smaller sub-regions until either their sub-regions' size reaches the predefined size, or sub-region includes a sufficiently small number of the non-overlapped object and then their division information is saved at a suitable search tree structure [3]. However, in the implementation method, they have a variety of subdivision methods and constructing methods of the search tree. In this thesis, I concentrate on the kd-tree acceleration scheme because it is the most widely used acceleration scheme.

The kd-tree is a search data structure utilized in ray tracing to search for an object, which has intersection with a ray, efficiently within a large number of objects. As depicted in Figure 7, the kd-tree builds a binary search tree based on topological information distributed in a multi-dimensional space and subsequently stores the reference information from which the target objects can be traced [5-7,28]. In this case, all objects in the kd-tree must be non-merged with each other. According to this restriction, if the system has merged objects (*i.e.*, CSG), then the kd-tree has to treat these merged objects as a single object (*i.e.*, the kd-tree does not subdivide the CSG object). This means if the system has many merged objects, consequently the efficiency of the kd-tree will be rapidly decreased.



**Figure 7.** Objects distribution in 2-D and related kd-tree structure.

In this section, I briefly introduced the kd-tree structure and its main functions. The kd-tree is a kind of binary space partitioning (BSP) tree which adaptively subdivide space into irregularly sized regions [3]. Thus, this scheme can be much more effective than a regular grid(or uniform grid) for irregular collections of geometry [3]. The kd-tree has two important functions. The first is a construction function, and the next is a traversal function.

The construction function starts with a bounding box that encompasses the entire system. If the number of objects in the box is greater than some threshold, the box is split in half by a plane. The objects are then assigned to whichever half they overlap. The objects, which lie in both halves, are assigned both parts twice. This process continues recursively until either each sub-region contains a sufficiently small number of objects, or a maximum splitting depth is reached. The splitting plane is restricted in a perpendicular plane to one of the coordinate axes [3]. These days many of the kd-tree schemes use a surface area heuristic (SAH) method instead of this simple subdivision method [7]. My improved kd-tree scheme also uses the SAH method, because this provides a more efficient search operation and a well-balanced binary search tree structure. The construction algorithm of the kd-tree likes this

---

**Algorithm 2.** Recursive KD-tree construction [5]

---

```
function RecBuild(objects O, bound box B) return node
    if Terminate(O, B) then
        return new leaf node(O)
    end if
    p = FindPlane(O, B) { Find a 'good' plane p to split B }
    (BL, BR) = Split B with p
    OL = { o ∈ O | (o ∩ BL) ≠ ∅ }
    OR = { o ∈ O | (o ∩ BR) ≠ ∅ }
    return new node(p, RecBuild(OL, BL), RecBuild(OR, BR))
end function

function BuildKDTree(objects[] O) return root node
    B = β(O) { start with full scene }
    return RecBuild(O, B)
end function
```

---

The 'good' condition is evaluated with the SAH cost function. The SAH cost function is called at every recursive call and, at that time, this function estimates a next traveling cost of a ray and then returns its cost value. There is detailed information about the SAH cost function in [5, 6, 42].

The traversal function starts with a ray and a root node of the kd-tree. The root node has a bounding box information of the entire system. At first, this checks intersection point between the ray and the bounding box. If there is no intersection between the ray and the bounding box, then this returns no intersection. If intersection exists, then this pushes I<sub>min</sub> point, I<sub>max</sub> point (see Figure 5(b)) and the root node reference into the stack and next pop the stack and evaluate whether or not the node is a leaf node. If the node is a leaf node, then finds intersection points between the ray and all objects inside the node and returns the nearest point. If the node is an internal node, then traverses left and right child of the node, and process same operation with each child node, until the stack is emptied. This traversal function is summarized like this

---

**Algorithm 3.** Traversal KD-tree

---

```
function TraversalKDTree(KD-tree root N, Ray R, Point &iP) return boolean  
an  
    (I_min, I_max) := intersect R with N's bounding box  
    if ray does not intersect bounding box then  
        iP := NULL  
        return false  
    end if  
    boolean bIntersect := false  
    push ( N, I_min, I_max) to stack  
    while stack is not empty do  
        (newNode, newI_min, newI_max) := pop stack  
        while newNode is not a leaf do  
            axis := newNode's split axis  
            tplane := (newNode split position[axis]  
                        - R::Initial[axis])/R::Direction[axis]  
            if tplane <= newI_min then  
                newNode := right child node  
                continue  
            end if  
            if tplane >= newI_max then  
                newNode := left child node  
                continue  
            end if  
            push (right child node, tplane, newI_max) to stack  
            newNode := left child node  
            newI_max = tplane  
        end while  
        if newNode is not empty then  
            intersect ray with all object inside node  
            if intersection exists then
```

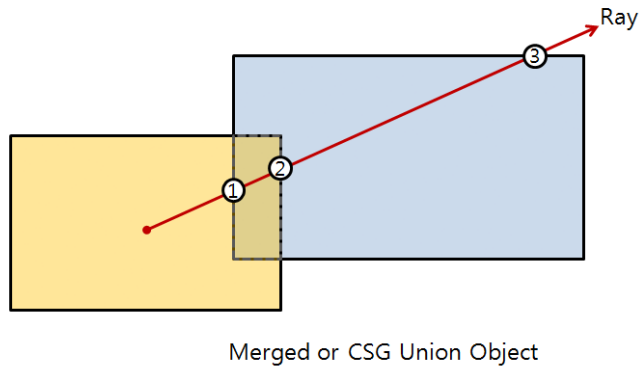
```

        iP = nearest intersection point
        bIntersect = true
    end if
end if
end while
return bIntersect
end function

```

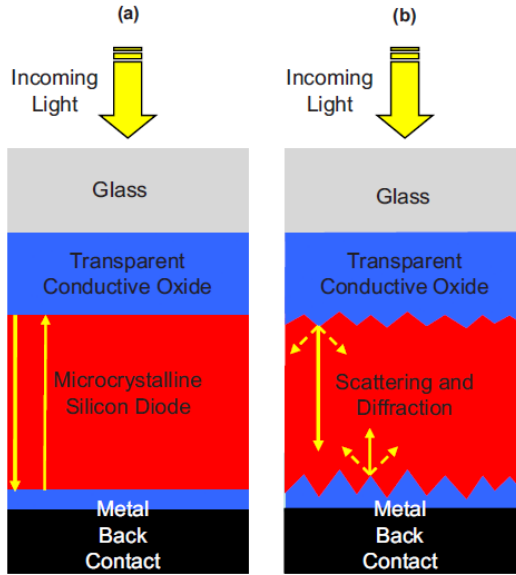
---

If a conventional traversal function of the kd-tree like Algorithm 3 is used, it cannot find an exact intersection point of merged objects (*e.g.*, CSG). Because merged objects have a more complex shape than their original shapes (*i.e.*, Some parts of surface of each object can be removed) so in some cases the nearest intersection point is not a valid intersection point of the merged objects. Such case is shown in Figure 8.



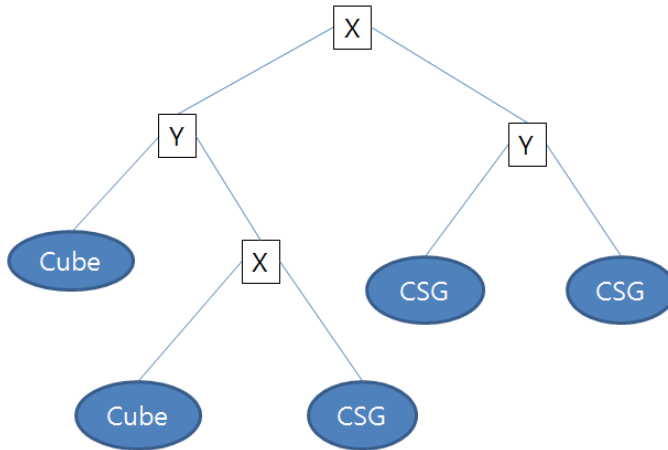
**Figure 8.** Scheme of conventional kd-tree traversal with CSG Object.

The conventional traversal algorithm will find point 1 as an intersection point. However, point 1 and point 2 are not valid intersection points but point 3 is valid. This is the reason why CSG objects cannot be treated as independent objects inside the kd-tree. For example, a textured silicon solar cell depicted in Figure 9(b) has three CSG objects and two primitive cube objects.



**Figure 9.** Schematic sketch of a thin-film microcrystalline silicon solar cell (a) on a smooth substrate and (b) on a randomly textured substrate [39]

Thus, if the conventional kd-tree scheme is used to analyze the textured silicon solar cell with the ray tracing method, Algorithm 2 constructs a kd-tree which is composed of only five objects (*i.e.*, Three CSG objects and two cube objects) like Figure 10.



**Figure 10.** Example of the kd-tree construction for textured silicon solar cells.

When Algorithm 3 reaches a leaf node which contains some of the five objects, it seeks intersection points of a ray and objects of the leaf node. At that time, the computational cost depends on the complexity of the leaf node object. If this object is a CSG object, then the cost is asymptotically  $O(N^2)$  for a number of  $N$  primitive elements of the CSG object [37]. In this case, the three CSG objects have a lot of pyramidal shaped primitives, so the intersection checking time for these kinds of CSG objects will rapidly increase.

There have been many researches [18-26] to decrease computation time for textured silicon solar cells. However, within my knowledge, there is no complete ray tracing solution for an arbitrarily textured silicon solar cell with an asymptotically average  $O(N_r \times N_i \times \log 2N_o)$  time complexity. In Chapter 2, I proposed a new ray tracing algorithm for an arbitrarily textured silicon solar cell with asymptotically average  $O(N_r \times N_i \times \log 2N_o)$  time complexity.

#### 1.4 Finite Difference Time Domain (FDTD)

If the thin film solar cell has a micrometer or nanometer sized parts, the ray tracing method cannot be used to analyze this system exactly, except for flat multi-layers. Because interferences, diffractions, and Plasmon effects are important electromagnetic phenomena in the sub-wavelength region. However, the ray tracing method cannot simulate these kinds of optical phenomena except for special cases. To solve this problem, Maxwell equation is needed, which are general governing equations for electromagnetic problems. In differential form, Maxwell equation is

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H} - \mathbf{J}, \quad (1.7)$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}, \quad (1.8)$$

$$\nabla \cdot \mathbf{D} = \rho, \quad (1.9)$$

$$\nabla \cdot \mathbf{B} = 0, \quad (1.10)$$

where  $\mathbf{H}$  is a magnetic field,  $\mathbf{E}$  is an electric field,  $\mathbf{D}$  is an electric flux density,  $\mathbf{B}$  is a magnetic flux density,  $\mathbf{J}$  is an electric current density, and  $\rho$  is a charge density. The bold characters mean vector values [1]. If no free electric charge,

linear, isotropic, and non-dispersive materials are assumed, Equation(1.7) - (1.10) can be rewritten like these

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon} \nabla \times \mathbf{H} - \frac{1}{\varepsilon} (\mathbf{J}_s + \sigma \mathbf{E}) , \quad (1.11)$$

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu} \nabla \times \mathbf{E} , \quad (1.12)$$

$$\nabla \cdot \mathbf{E} = 0 , \quad (1.13)$$

$$\nabla \cdot \mathbf{H} = 0 , \quad (1.14)$$

where  $\varepsilon$  is an electric permittivity,  $\mu$  is a magnetic permeability,  $\sigma$  is an electric conductivity, and  $\sigma^*$  is a magnetic conductivity.  $\mathbf{J}$  can be thought of independent sources of  $\mathbf{E}$  field energy  $\mathbf{J}_s$  and dependent sources of  $\mathbf{E}$  field energy  $\sigma \mathbf{E}$  [13]. In Cartesian coordinates, the vector components of the curl operators of Equation(1.11) and (1.12) can be written as

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - (J_{s_x} + \sigma E_x) \right] , \quad (1.15)$$

$$\frac{\partial E_y}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - (J_{s_y} + \sigma E_y) \right] , \quad (1.16)$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon} \left[ \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - (J_{s_z} + \sigma E_z) \right] , \quad (1.17)$$

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \right] , \quad (1.18)$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \right] , \quad (1.19)$$

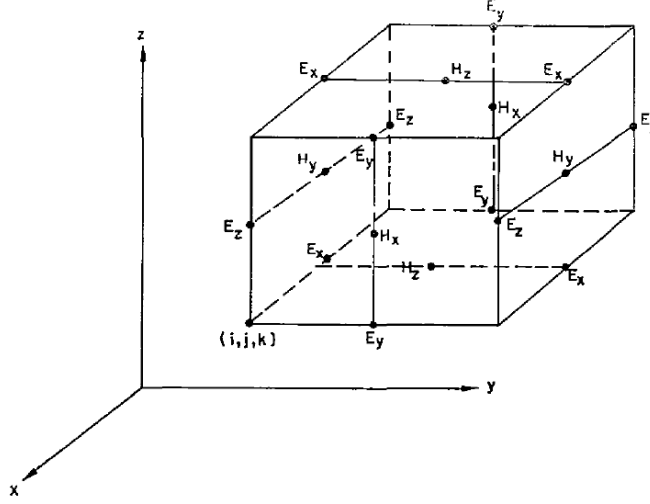
$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu} \left[ \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \right] . \quad (1.20)$$

These six coupled partial differential equations of Equation(1.15) - (1.20) are governing equations of electromagnetic problems in 3-D spaces. A FDTD numerical algorithm starts from these equations. In no free charge, Equation(1.13) and (1.14) are satisfied naturally, so these equations need not explicitly appear in the FDTD algorithm [13].

In 1966, K. S. Yee introduced a set of finite difference equations to the curl equations of Equation (1.15) - (1.20) for the lossless materials [14]. Yee defined a problem's spatial domain as a cubical shape and discretized sub-cubical spaces and then assigned E- and H-fields as Figure 11. The E- and H-fields stagger each other in Figure 11. Yee also used central difference scheme



to evaluate space and time derivatives. Using this scheme, he obtained second order accuracy.



**Figure 11.** Positions of various field components. The E-fields are in the middle of the edges and the H-fields are in the center of the faces [14]

According to the Yee scheme, Equation(1.15) is converted to a following finite difference equation

$$\frac{E_x|_{i+\frac{1}{2},j,k}^{n+1} - E_x|_{i+\frac{1}{2},j,k}^n}{\Delta t} = \frac{1}{\epsilon_{i+\frac{1}{2},j,k}} \left[ \frac{H_z|_{i+\frac{1}{2},j+\frac{3}{2},k}^{n+\frac{1}{2}} - H_z|_{i+\frac{1}{2},j+\frac{1}{2},k}^{n+\frac{1}{2}}}{\Delta y} - \frac{H_y|_{i+\frac{1}{2},k+\frac{3}{2}}^{n+\frac{1}{2}} - H_y|_{i+\frac{1}{2},k+\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta z} - J_{sx}|_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}} - \sigma_{i+\frac{1}{2},j,k} E_x|_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}} \right] \quad (1.21)$$

Note that all E-field values have a  $n+1$  or  $n$  time steps in the FDTD calculation, so  $E_x|_{i,j+1/2,k+1/2}^{n+1/2}$  value cannot come from a pre-calculated value directly.

To solve this problem, he used simple semi-implicit approximation

$$E_x|_{i,j+1/2,k+1/2}^{n+1/2} = \frac{E_x|_{i,j+1/2,k+1/2}^{n+1} + E_x|_{i,j+1/2,k+1/2}^n}{2}. \quad (1.22)$$

The other Equation(1.16) - (1.20) are also converted similar way [13].

### 1.4.1 Discretization of the System Domain

The first step of the FDTD method is to define the computational domain of the targeting system. This computational domain can include all the system or only sub region of the system. I assumed Cartesian coordinate aligned cubical boundary in the 3-D case and rectangle boundary in the 2-D case according to the Yee scheme [14]. The domain discretization is to divide the FDTD system domain into smaller sub-domains. If the subdivision delta or cell size is determined by the system geometry and the used wavelength, a suitable maximum time step is determined by Courant number [13]. In one dimensional case, Courant number is like this

$$S \equiv \frac{c\Delta t}{\Delta x} \leq 1, \Delta t \leq \frac{\Delta x}{c}, \quad (1.23)$$

where S is Courant number, c is the free-space speed of light,  $\Delta t$  is time step, and  $\Delta x$  is cell size [13].

The Courant number is a necessary condition to obtain numerical stability. Equation(1.23) is a maximum number which can be selected. If some complex system is needed to simulate, which has curved shape geometry or heterogeneous material, a smaller Courant number has to be chosen for numerical stability.

With the discretization conditions, the E- and H-Field update equations can be derived from Equation(1.21) with the assumption of input source free and lossless condition like this

$$E_x|_{i+\frac{1}{2},j,k}^{n+1} = E_x|_{i+\frac{1}{2},j,k}^n + \frac{\Delta t}{\varepsilon_{i+\frac{1}{2},j,k}} \left[ \frac{H_z|_{i+\frac{1}{2},j+\frac{3}{2},k}^{n+\frac{1}{2}} - H_z|_{i+\frac{1}{2},j+\frac{1}{2},k}^{n+\frac{1}{2}}}{\Delta y} - \frac{H_y|_{i+\frac{1}{2},j,k+\frac{3}{2}}^{n+\frac{1}{2}} - H_y|_{i+\frac{1}{2},j,k+\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta z} \right], \quad (1.24a)$$

$$H_x|_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n+\frac{3}{2}} = H_x|_{i,j+\frac{1}{2},k+\frac{1}{2}}^{n+\frac{1}{2}} + \frac{\Delta t}{\mu_{i,j+\frac{1}{2},k+\frac{1}{2}}} \left[ \frac{E_y|_{i,j+\frac{1}{2},k+1}^n - E_y|_{i,j+\frac{1}{2},k}^n}{\Delta z} - \frac{E_z|_{i,j+1,k+\frac{1}{2}}^n - E_z|_{i,j,k+\frac{1}{2}}^n}{\Delta y} \right] \quad (1.24b)$$

### 1.4.2 Dispersive Materials

A dispersive material has different electric permittivity and magnetic permeability values according to the affected field frequency or the field intensity. Most of the materials used in the thin film solar cells are the dispersive material.

In this these, I consider only linear dispersion of which optical constants are varied by the frequency of a field and not by the field intensity. The linear dispersion can be modelled by Drude, Lorentz, and Drude-Lorentz models [13, 73, 74]. These models can be unified to the Lorentz model with a simple assumption.

#### 1.4.2.1 Lorentz Model

The Lorentz model is used to describe the optical property of a general linear dispersive material. It assumes that the electrons and atoms of the material are bounded each other like a spring. The frequency-domain susceptibility function can be written as

$$\chi(\omega) = \sum_{p=1}^N \frac{\Delta\epsilon_p \omega_p^2}{\omega_p^2 - \omega^2 - 2j\omega\delta_p} , \quad (1.25)$$

where  $\Delta\epsilon_p$  is the change in relative permittivity due to the Lorentz pole pair,  $\omega$  is the angular frequency,  $\omega_p$  is the plasma frequency of the Lorentz pole pair,  $N$  is the number of pole pair, and  $\delta_p$  is the damping coefficient. Therefore, frequency-domain permittivity function can be written as

$$\epsilon(\omega) = \epsilon_\infty + \sum_{p=1}^N \frac{\Delta\epsilon_p \omega_p^2}{\omega_p^2 - \omega^2 - 2j\omega\delta_p} , \quad (1.26)$$

where  $\epsilon_\infty$  is a permittivity value when the wavelength value goes to infinity.

Equation(1.25) and (1.26) are frequency-domain functions, so these cannot be applied to the FDTD system directly. To obtain the time-domain functions, mainly two kinds of solutions were proposed. The first is the piecewise-linear recursive convolution (PLRC) method and the second is the auxiliary differential equation (ADE) method [13, 77]. I used only the ADE method in this thesis.

To apply the ADE method, I started with Ampere's law, which is written as

$$\nabla \times \vec{H}(t) = \epsilon_0 \epsilon_\infty \frac{d}{dt} \vec{E}(t) + \sigma \vec{E}(t) + \sum_{p=1}^N \vec{J}_p(t) , \quad (1.27)$$

where  $\vec{J}_p(t)$  is the polarization current associated with the  $p$ -th Lorentz pole pair [13]. In this equation, Phasor polarization current is

$$\vec{J}_p(\omega) = \epsilon_0 \Delta\epsilon_p \omega_p^2 \left[ \frac{j\omega}{\omega_p^2 - \omega^2 - 2j\omega\delta_p} \right] \vec{E}(\omega) \quad (1.28)$$

If  $(\omega_p^2 - \omega^2 + 2j\omega\delta)$  is multiplied to the both side of Equation(1.28), then the result is

$$\omega_p^2 \check{\mathbf{J}}_p(\omega) - 2j\omega\delta_p \check{\mathbf{J}}_p(\omega) - \omega^2 \check{\mathbf{J}}_p(\omega) = \varepsilon_0 \Delta \varepsilon_p \omega_p^2 j\omega \check{\mathbf{E}}(\omega) \quad (1.29)$$

Equation(1.29) can be easily converted to the time-domain equation using these equations

$$\frac{d}{dt} = (-j)\omega, \quad \frac{d^2}{dt^2} = -\omega^2, \quad (1.30)$$

With Equation(1.29) and Equation(1.30),  $\vec{\mathbf{J}}_p(t)$  can be achieved as follow :

$$\omega_p^2 \vec{\mathbf{J}}_p(t) + 2\delta_p \frac{d}{dt} \vec{\mathbf{J}}_p(t) + \frac{d^2}{dt^2} \vec{\mathbf{J}}_p(t) = \varepsilon_0 \Delta \varepsilon_p \omega_p^2 \frac{d}{dt} \vec{\mathbf{E}}(t) \quad (1.31)$$

Equation(1.31) is the ADE for  $\vec{\mathbf{J}}_p(t)$ . Therefore, a update equation for  $\vec{\mathbf{J}}_p(t)$

can be made like this :

$$\omega_p^2 \vec{\mathbf{J}}_p^n + 2\delta_p \left[ \frac{\vec{\mathbf{J}}_p^{n+1} - \vec{\mathbf{J}}_p^{n-1}}{2\Delta t} \right] + \left[ \frac{\vec{\mathbf{J}}_p^{n+1} - 2\vec{\mathbf{J}}_p^n + \vec{\mathbf{J}}_p^{n-1}}{(\Delta t)^2} \right] = \varepsilon_0 \Delta \varepsilon_p \omega_p^2 \left[ \frac{\vec{\mathbf{E}}^{n+1} - \vec{\mathbf{E}}^{n-1}}{2\Delta t} \right], \quad (1.32)$$

$$\vec{\mathbf{J}}_p^{n+1} = \alpha_p \vec{\mathbf{J}}_p^n + \xi_p \vec{\mathbf{J}}_p^{n-1} + \gamma_p \left[ \frac{\vec{\mathbf{E}}^{n+1} - \vec{\mathbf{E}}^{n-1}}{2\Delta t} \right], \quad (1.33)$$

where

$$\alpha_p = \frac{2 - \omega_p^2 (\Delta t)^2}{1 + \delta_p \Delta t}, \quad (1.33a)$$

$$\xi_p = \frac{\delta_p \Delta t - 1}{1 + \delta_p \Delta t}, \quad (1.33b)$$

$$\gamma_p = \frac{\varepsilon_0 \Delta \varepsilon_p (\Delta t)^2 \omega_p^2}{1 + \delta_p \Delta t}. \quad (1.33c)$$

To evaluate  $\vec{\mathbf{E}}^{n+1}$  in Equation(1.27),  $\vec{\mathbf{J}}_p^{n+1/2}$  is needed. This value can be calculated as follow

$$\vec{\mathbf{J}}_p^{n+1/2} = \frac{1}{2} (\vec{\mathbf{J}}_p^n + \vec{\mathbf{J}}_p^{n+1}) = \frac{1}{2} \left[ (1 + \alpha_p) \vec{\mathbf{J}}_p^n + \xi_p \vec{\mathbf{J}}_p^{n-1} + \gamma_p \left[ \frac{\vec{\mathbf{E}}^{n+1} - \vec{\mathbf{E}}^{n-1}}{2\Delta t} \right] \right] \quad (1.34)$$

If Equation(1.31) is rewritten to a update equation for  $\vec{\mathbf{E}}^{n+1}$  and apply Equation(1.34) to the update equation then the last equation can be obtained like this :

$$\vec{\mathbf{E}}^{n+1} = C_1 \vec{\mathbf{E}}^{n-1} + C_2 \vec{\mathbf{E}}^n + C_3 \left\{ \nabla \times \vec{\mathbf{H}}^{n+1/2} - \frac{1}{2} \sum_{p=1}^N [(1 + \alpha_p) \vec{\mathbf{J}}_p^n + \xi_p \vec{\mathbf{J}}_p^{n-1}] \right\}, \quad (1.35)$$

where

$$C_1 = \frac{\frac{1}{2} \sum_{p=1}^N \gamma_p}{2\varepsilon_0 \varepsilon_\infty + \frac{1}{2} \sum_{p=1}^N \gamma_p + \sigma \Delta t}, \quad (1.35a)$$

$$C_2 = \frac{2\varepsilon_0\varepsilon_\infty - \sigma\Delta t}{2\varepsilon_0\varepsilon_\infty + \frac{1}{2}\sum_{p=1}^N \gamma_p + \sigma\Delta t}, \quad (1.35b)$$

$$C_3 = \frac{2\Delta t}{2\varepsilon_0\varepsilon_\infty + \frac{1}{2}\sum_{p=1}^N \gamma_p + \sigma\Delta t}. \quad (1.35c)$$

Equation(1.35) is the E-field update equation for dispersive material [13].

#### 1.4.2.2 Drude Model

The Drude model is used to describe the optical property of metal. It assumes that the optical properties of the metal are mainly affected by free electrons of the metal. The frequency-domain susceptibility function of the Drude model can be written as

$$\chi(\omega) = \frac{\Omega_p}{\omega^2 - 2j\omega\delta}, \quad (1.36)$$

where  $\omega$  is the angular frequency,  $\Omega_p$  is the plasma frequency of material and  $\delta$  is the collision frequency. Therefore, frequency-domain permittivity function can be written as

$$\varepsilon(\omega) = \varepsilon_\infty + \frac{\Omega_p}{\omega^2 - 2j\omega\delta}, \quad (1.37)$$

where  $\varepsilon_\infty$  is the permittivity value when the wavelength value goes to infinity.

Equation(1.36) and (1.37) are similar to Equation(1.25) and (1.26), respectively. If single Lorentz pole pair is assumed and  $\Omega_p = \Delta\varepsilon_p\omega_p^2$  and  $\omega_p$  goes to zero, then Equation(1.36) and (1.37) can be replaced with Equation(1.25) and (1.26), respectively. Therefore, I used same E-field and J-field update Equation(1.34) and (1.35), respectively, to the Drude model for unified solution.

#### 1.4.2.3 Drude-Lorentz Model

Most of the noble metals (*e.g.*, Gold, Silver, ...) have optical properties of metal and dielectric material both at less than 1000nm wavelength. To simulate these kinds of metals, the Drude-Lorentz model was proposed by A.D. Rakic et al. [73, 74]. The frequency-domain susceptibility function can be written as

$$\chi(\omega) = \frac{\Omega_p}{\omega^2 - 2j\omega\delta} + \sum_{p=1}^N \frac{\Delta\varepsilon_p\omega_p^2}{\omega_p^2 - \omega^2 - 2j\omega\delta_p}, \quad (1.38)$$

where  $\Delta\epsilon_p$  is the change in relative permittivity due to the Lorentz pole pair,  $\omega$  is the angular frequency,  $\Omega_p$  is the plasma frequency of material,  $\omega_p$  is the plasma frequency of the Lorentz pole pair,  $N$  is the number of pole pairs, and  $\delta$  is the damping coefficient. Therefore, frequency-domain permittivity function can be written as

$$\epsilon(\omega) = \epsilon_{\infty} + \frac{\Omega_p}{\omega^2 - 2j\omega\delta} + \sum_{p=1}^N \frac{\Delta\epsilon_p \omega_p^2}{\omega_p^2 - \omega^2 - 2j\omega\delta_p}, \quad (1.39)$$

where  $\epsilon_{\infty}$  is the permittivity value when the wavelength value goes to infinity. Equation(1.38) and (1.39) also can be unified to Equation(1.25) and (1.26), respectively.

I used this FDTD techniques to analyze the absorption energy of each layer within the thin film solar cells, and this kind of analysis executed with the ray tracing method when the thin film solar cells are composed of all plane layers.

### 1.4.3 Boundary Condition

When a physical system is analyzed by the FDTD method, it's all system domain can be modelled with a FDTD system. However, in many case, a part of the physical system domain is modelled with the FDTD system, and also if the physical system has infinite domain, the FDTD method cannot model the entire physical system domain. Thus, the FDTD method needs a suitable computational boundary condition which can mimic the entire physical system domain.

I introduced two kinds of computational boundary conditions for the FDTD method in the next sections. The first is an absorbing boundary condition, and the second is a periodic boundary condition.

#### 1.4.3.1 Absorbing Boundary Condition (ABC)

If a radar radiation pattern problem is needed to simulate with the FDTD method, the problem has to be solved under the far field radiation condition. The far field radiation condition means that the radiated field energy approaches to zero when it goes to infinite. The absorbing boundary condition (ABC) can be used to solve these kinds of problems, such that it makes any reflected field energies from the FDTD system boundary to be zero [125, 126].

The ABC can be categorized with global ABC and local ABC. The global ABCs based on the fact that the field at any point on the boundary of the FDTD system can be expressed as a superposition of the fields of all the sources at retarded-time. However, this needs much computational time and resources. The local ABC treats the field on the boundary as a field function in the vicinity of the considered point [128].

B. Enquist and A.Majda [125] analytically solved the local ABC by one-way travelling wave equation with first, second, and third order accuracies as in Equation [125]-(1,2,3). These analytical solutions were implemented by G. Mur [126]. Thus, this ABC is highly effective when the absorbed wave is travelling waves. However, this is not suitable for evanescent waves.

At the 1993, J. P. Berenger proposed a new local ABC which is called a perfectly matched layer (PML) ABC. This assumes a nonphysical absorbing medium that has a perfectly matched impedance at computational boundary interfaces. This medium is placed in the outer FDTD cells of the computational domain to absorb the outgoing waves. In this matched medium, the Maxwell equations are replaced with [128]

$$\epsilon_0 \frac{\partial \vec{E}}{\partial t} + \sigma \vec{E} = \nabla \times \vec{H}, \quad (1.40a)$$

$$\mu_0 \frac{\partial \vec{H}}{\partial t} + \sigma^* \vec{H} = -\nabla \times \vec{E}, \quad (1.40b)$$

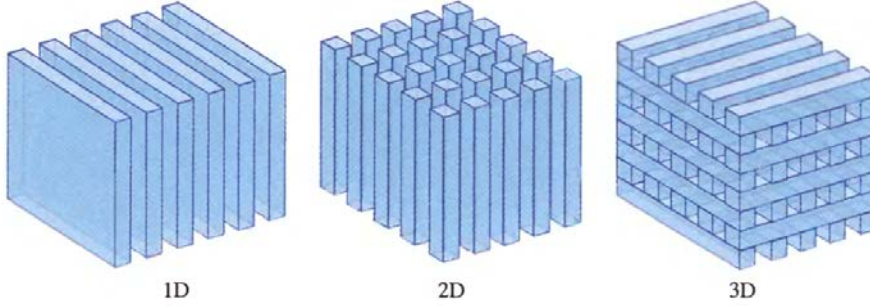
where  $\sigma^*$  is a nonphysical parameter that allows the absorption of the magnetic field to be symmetrised with respect to the absorption of the electric field, provided that the following matching condition :

$$\frac{\sigma}{\epsilon_0} = \frac{\sigma^*}{\mu_0} \quad (1.41)$$

The impedance of a plane wave in the medium equals the impedance in a vacuum [128]. This PML ABC theory was implemented by Split PML[127, 129], Uniaxial PML (UPML)[130], and Convolutional PML (CPML) [131] in the time domain for the FDTD method. In this thesis, I used the CPML technique to implement the ABC.

### 1.4.3.2 Periodic Boundary Condition (PBC)

The PBC is other kinds of boundary condition that treats a one-dimensional, two-dimensional, and three-dimensional periodic structures like Figure 12. For example, a photonic band gap (PBG) structure has 2-D periodic structure.

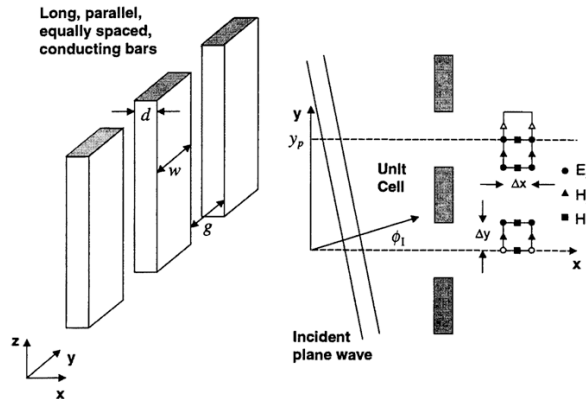


**Figure 12.** Periodic structures in one-dimensional (1-D), two-dimensional (2-D), and three-dimensional (3-D) configurations [1].

As an example, consider two-dimensional electromagnetic screen in Figure 13. The screen consists of infinitely long, parallel, conducting bars of width  $w$ , thickness  $d$ , and separated by a gap  $g$ . The illumination is an electromagnetic plane wave with the electric field parallel to the bars, and the propagation direction at angle  $\phi_I$  with respect to the  $x$ -axis. In this condition, the time domain boundary condition is

$$H_x\left(x, y = y_p + \frac{\Delta y}{2}, t\right) = H_x\left(x, y = \frac{\Delta y}{2}, t - y_p \sin \phi_I\right), \quad (1.42a)$$

$$E_z(x, y = 0, t) = E_z(x, y = y_p, t + y_p \sin \phi_I). \quad (1.42b)$$



**Figure 13.** Two-dimensional electromagnetic screen (left) and a top view of the structure [13].



If the incident angle  $\theta_i$  of the plane wave is zero (*i.e.*, normal incident), then Equation 1.42 are rewritten like this

$$H_x\left(x, y = y_p + \frac{\Delta y}{2}, t\right) = H_x\left(x, y = \frac{\Delta y}{2}, t\right), \quad (1.43a)$$

$$E_z(x, y = 0, t) = E_z(x, y = y_p, t). \quad (1.43b)$$

Therefore, for the normal incident plane wave, the problem can be easily solved with the current time field values at the both boundary interfaces. However, if the impinging plane wave has not normal incident angle, the PBC problems cannot easily treated because the broadband oblique incident wave impinging in the dispersive medium suffers time delay or advance during propagation. To solve this kind of problems, many techniques were proposed, and more information of the techniques is explained in Taflove et al. [13].

## 1.5 Scope and Objectives

I researched new numerical methods to analyze the enhanced absorption energy of the thin film solar cells precisely and efficiently. Specifically, my research concentrated on developing a novel ray tracing algorithm to analyze 3-D textured thin film solar cells and a new FDTD method to simulate nanometer size particles immersed or nanometer scale thin film solar cells, respectively.

## 1.6 Achievements

Through this research, I developed three kinds of algorithms. The first is the Slab-Outline algorithm which dramatically reduces the collision detection time when some layer of the thin film solar cell has an enormous number of embedded particles on its surface or inside it. The second is a direct calculation algorithm to simulate enhanced absorption energy inside the thin film solar cells with the ray tracing method. The Third is an object's boundary auto-detection and an absorption energy calculation algorithm with the FDTD method.

## 2 Slab-Outline Algorithm for Fast Intersection Finding

### 2.1 Overview

In this chapter, I propose a novel intersection algorithm for 3-D texture geometries, which can precisely find an intersection point with an  $O(\log N)$  average time complexity. This complexity is superior to other algorithms' one in this case.

Since the first implementation of pyramidal texturing into a silicon(Si) solar cells [15], the three-dimensional (3-D) texturing of a silicon surface together with an antireflection treatment of the surface have been widely utilized to enhance the confinement of the incoming light in Si solar cells [16, 17, 31-35]. Figure 9 shows a general structure of textured thin-film microcrystalline Si solar cells. Optimizing the design of the surface texture is an essential aspect of the thin film Si solar cells technology as it can maximize light trapping efficiency of the cells [12]. Numerical modeling and an analytical solution can provide an efficient and meaningful pathway in designing an optimized texture pattern. If the investigation procedure is done solely by experimental procedures without these numerical solutions, this involves expensive and time-consuming work [12].

The ray tracing method is a very powerful technique that can be used to fulfill the above purpose. However, earlier numerical tools based on the ray tracing method are associated with limitations when simulating actual texture patterns because they can handle only a regularly sized pyramidal pattern or a regular array of a regular pattern. They also require a long execution time [18-21]. Some algorithms utilized only a unit-cell to explain the experimental result, and others calculated the total absorption energy inside the solar cell system indirectly by obtaining the total reflectance and transmittance without considering the interference [22, 23]. In later research, two additional advanced ray tracing techniques in which randomly sized pyramidal shapes and patterns could be handled to some extent were developed [24, 25]. In these

methods, one had a restriction related to the positioning of adjacent unit cells in a certain range of around the target unit cell, through which finding the adjacent unit cells becomes feasible during ray tracing, resulting in a failure to handle truly random patterns [24]. The other technique utilized premeasured topological information of real surfaces (i.e., 2-D position and height mapping) obtained from measurements by atomic force microscopy (AFM) during a simulation of ray tracing. However, this technique also had difficulty in generating various patterns for the simulation [25]. Furthermore, both above methods used asymptotic  $O(N_r \times N_i \times N_o)$  time complexity, leading to limitations in increasing the system size and number of rays. In asymptotic time complexity,  $N_r$ ,  $N_i$  and  $N_o$  indicate the number of rays, the average number of intersections, and the number of objects, respectively [12]. It was also shown that a reduced computing time could be attained using pixel approximation [26], however, the algorithm aimed at module simulation by partitioning the module into many segments and assuming a beam instead of tracing rays individually [12].

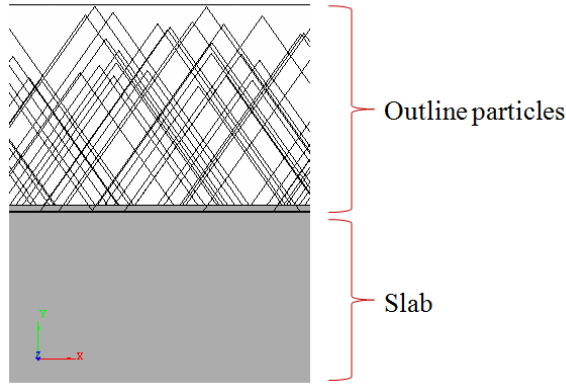
In this chapter, I propose a newly devised algorithm termed '*Slab-Outline*' which is based on the kd-tree. It can handle a fully random texture in terms of the shape, size, and distribution of a 3-D pattern. The *Slab-Outline* algorithm can trace the actual object geometry with asymptotic  $O(N_r \times N_i \times \log N_o)$  time complexity [12]. This algorithm also can be used to make photorealistic 3-D images and to analyze general purpose optical systems. Furthermore, this can solve the hole or immerse type texturing problems. The efficiency of the computing time and the validity of the *Slab-Outline algorithm* were tested and proved using 'Raywiz-SOLAR' made by InsideOptics, Inc., a tool that contains the *Slab-Outline algorithm* [27].

## 2.2 Algorithm

The *Slab-Outline* algorithm uses an improved kd-tree traversal algorithm and a new idea of separation of the substrate slab from the pyramid outline patterns. With these techniques, the algorithm can precisely solve the ray tracing problem of the randomly textured silicon solar cells, that are composed of a

lot of embedded pyramidal patterns, with asymptotically  $O(N_r \times N_i \times \log N_o)$  average time complexity [12].

The improved kd-tree has the same data structure and functions of the conventional kd-tree except for the traversal function. This modified function allows to traverse the textured geometry with the same time complexity of the conventional kd-tree exactly. The idea of separation of the outline particles from the substrate slab treats the pyramidal textured complex object as separated two kinds of objects, slab and outline particles like Figure 14.

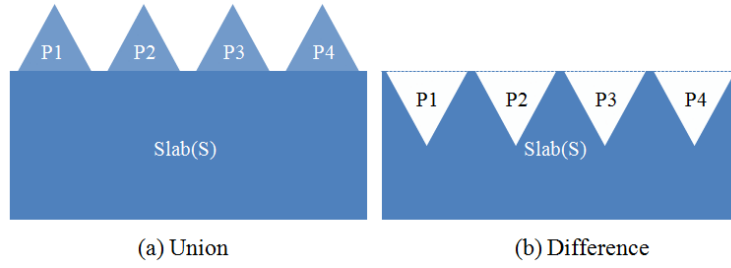


**Figure 14.** Cross section view of 3-D textured solar cell.

For clear understanding of the Slab-Outline algorithm, I divided this problem into two cases. One is that the system is composed of non-overlapped pyramidal textures and substrate slab, and the other is the system is composed of overlapped pyramidal textures and substrate slab. The second is more general case but more difficult to solve.

### 2.2.1 Non-overlapped texture case

Like Figure 15 shows, this type of textured silicon solar cell has non-overlapped pyramidal patterns, but all these patterns have merged or differed region with the substrate slab.



**Figure 15.** Schema for non-overlapped texture of silicon solar cells. (a) is normal pyramidal shape, and (b) is inverse pyramidal shape.

The two shapes of Figure 15 can be made by CSG union and difference objects. Such that (a) is a CSG union and (b) is a CSG difference, respectively. The conventional CSG algorithm treats all primitive objects equally and requires  $O(N^2)$  time complexity because it assumes all primitives will participate in CSG operation each other. The CSG equations like these

$$\{ S \cup P_1 \cup P_2 \cup P_3 \dots \cup P_{n-1} \} = (a) , \quad (2.1)$$

$$\{ S - P_1 - P_2 - P_3 \dots - P_{n-1} \} = (b) . \quad (2.2)$$

However, in this case, I assumed non-overlapped pyramidal texture, so all pyramidal primitives do not participate in the CSG operation with each other except for they participate in CSG operation with the substrate slab. These modified CSG equations like these

$$\{ S \cup \{P_1, P_2, P_3 \dots, P_{n-1}\} \} = (a) , \quad (2.3)$$

$$\{ S - \{P_1, P_2, P_3 \dots, P_{n-1}\} \} = (b) , \quad (2.4)$$

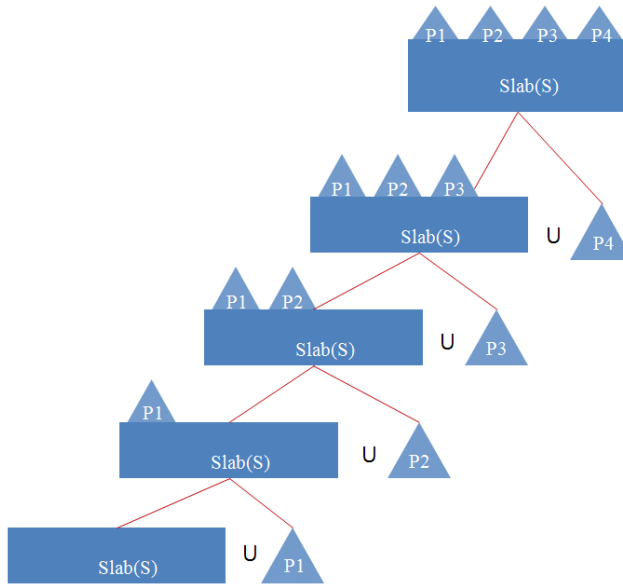
$$\{ P_i \cap P_j \} = \emptyset, \forall (i \neq j). \quad (2.5)$$

The Slab-Outline algorithm starts with Equation(2.3)-(2.5). According to these equations, new CSG notations can be defined for these cases. The conventional CSG notations likes these

$$\text{Object (a)} \equiv \text{union} \{ \text{Object S, Object } P_1, \text{Object } P_2, \dots, \text{Object } P_{n-1} \} , \quad (2.6)$$

$$\text{Object (b)} \equiv \text{difference} \{ \text{Object S, Object } P_1, \text{Object } P_2, \dots, \text{Object } P_{n-1} \} . \quad (2.7)$$

Equation(2.6)-(2.7) are pseudo scripts for Equation(2.1)-(2.2), from these notations, the ray tracing program constructs a CSG tree and operation sequences like Figure 16.



**Figure 16.** Example of CSG tree and operation sequences.

Meanwhile, my new CSG notations for the Slab-Outline algorithm likes these

$$\text{Object } S' \equiv \text{Object } S + \text{slab flag} , \quad (2.8)$$

$$\text{Object } P'(n) \equiv \text{Object } P(n) + \text{outline flag} , \quad (2.9)$$

$$\text{Object } (a|b') \equiv \text{Object } (a|b) + \text{slab-outline flag} , \quad (2.10)$$

Object (a')  $\equiv$

slab-outline union { Object S', union {Object P'\_1, Object P'\_2, ..., Object P'\_{n-1}} } , (2.11)

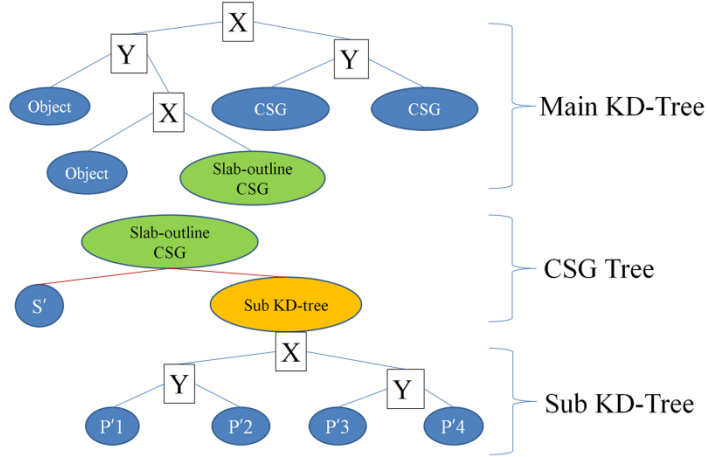
Object (b')  $\equiv$

slab-outline difference { Object S', union {Object P'\_1, Object P'\_2, ..., Object P'\_{n-1}} } . (2.12)

Equation(2.8)-(2.12) are new notations for Equation(2.3)-(2.5). Equation (2.8)-(2.10) indicate that each object class has additional flags to distinguish between a slab-outline object and a normal object. The 'union' operator inside the brace indicates that the grouped P' objects must be separated from Object S', and the P' objects are thought as a single merged object. In the non-overlapped case, these P' objects do not have any overlapped region with each other. This indicates that all P' objects' surfaces construct a single merged surface without loss.

With these new CSG notations and properties, The kd-tree of Figure 10 can be rebuilt with three sub kd-trees, which contain only the relevant infor-

mation pertaining to the patterns or particles that form the topological texture of the surface and the surrounding media, including the substrate slab are combined with a new CSG operation. The modified structure of Figure 10 is described in Figure 17.



**Figure 17.** Modified kd-tree with the Slab-Outline algorithm.

To treat this modified kd-tree, at first, I defined new data types. These are 'Object', 'MaterialProperty', and 'CSG'. The 'Object' is a class for primitive object as follow :

---

**Class 1. new Object**

---

**Class Object**

**MaterialProperty MP**

**int** flag { none | slab | outline }

**end Class**

---

where the 'MaterialProperty' is a class for material property of an object which is defined as

---

**Class 2. Structure of material property of Object**

---

**Class MaterialProperty**

**int** flag { non-dispersive | dispersive }

**float**  $\epsilon[3]$

**float**  $\sigma[3]$

---

**end Class**

---

where flag is an indicator of dispersive or non-dispersive,  $\epsilon[3]$  is electric permittivity of an object in x, y, and z axes respectively, and  $\sigma[3]$  is conductivity of an object in x, y, and z axes respectively. In simulation, I assumed the magnetic permeability is  $\mu_0$ , i.e., magnetic permeability is same as in vacuum.

The 'CSG' is a class to represent the primitive objects that are participated in CSG operation likes this :

---

**Class 3. new CSG Object**

---

**Class CSG**

**int** type { union | difference | intersection | immerse }

**Object[]** reference elements

**KD-Tree** reference subtree

**boolean** slab-outline flag

**end Class**

---

Next, I developed a new CSG construction algorithm that also is used in the overlapped pyramidal texture case in the next section. The construction algorithm is shown as follow

---

**Algorithm 4. Construct CSG object**

---

**function** ConstructCSG(String CSG\_script, Object[] primitives) **return** CSG object

**if** GetToken(script, "slab-outline") **then**

    pCSG := Create CSG class instance

    pCSG::elements := FindSlabObject(Object[] primitives)

    pCSG::subtree :=

        BuildKDTree({primitives - elements[0]}) from Algorithm 2.

    pCSG::slab-outline := true

**return** pCSG

**else**

**return** Construct conventional CSG object



**end if**  
**end function**

---

Then, I defined new traversal algorithms for the sub kd-tree and the Slab-Outline CSG object. This is the most important part of the Slab-Outline algorithm. If a ray meets the Slab-Outline CSG object during its tracing process within the main kd-tree, the intersection points encountered by the proceeding ray are obtained by checking both the sub kd-tree and the slab object. The new traversal algorithms for the Slab-Outline CSG object like these :

---

**Algorithm 5.** Traversal Sub KD-tree

---

**function** TraversalSubKDTree(KD-tree root N, Ray R, *Point[] iP, Object &IObject*) **return** boolean

(I\_min, I\_max) := intersect R with N's bounding box

**if** ray does not intersect bounding box **then**

iP := NULL

**return** false

**end if**

boolean bIntersect := false

**push** ( N, I\_min, I\_max) **to stack**

**while** stack is not empty **do**

(newNode, newI\_min, newI\_max) := **pop stack**

**while** newNode is not a leaf **do**

axis := newNode's split axis

tplane := (newNode split position[axis]

- R::Initial[axis])/R::Direction[axis]

**if** tplane <= newI\_min **then**

newNode := right child node

**continue**

**end if**

**if** tplane >= newI\_max **then**

newNode := left child node

```

        continue
    end if
    push (right child node, tplane, newI_max) to stack
    newNode := left child node
    newI_max = tplane
end while
if newNode is not empty then
    intersect ray with all object inside node
    if intersection exists then
        iP := intersection points of nearest object
        IObject := nearest object
        bIntersect := true
    end if
end if
return bIntersect
end function

```

---

Algorithm 5 is an improved version of Algorithm 3 which is used to traverse the conventional kd-tree. The difference of the two algorithms is their return parameters and check logic for intersection points. Algorithm 3 returns only the nearest intersection point and does not return the intersected object. However, Algorithm 5 returns set of intersection points with the intersected object. The object and the points will be used by Algorithm 6 and Algorithm 7. The colored italic font is used to describe the modified regions.

---

**Algorithm 6.** Traversal of Slab-Outline CSG Union object

---

```

function TraversalCSGUnion (Object CSG, Ray R, Point &iP) return boolean
if CSG is slab-outline then
    if TraversalPrimitive(CSG::elements[0], R, slabIntersects[]) then
        // intersect ray with slab object
    end if
end if

```

```

[P1] :   if TraversalSubKDTree(CSG::subtree, R, outlineIntersects[], IObject)
then
    // intersect ray with outline object
    while slabIntersects[] is not empty do
        if slabIntersects[i] doesn't exist in IObject then
            iP := min(iP, slabIntersects[i])
        end if
        i := i+1
    end while
    while outlineIntersects[] is not empty do
        if outlineIntersects[j] doesn't exist in CSG::element[0] then
            if outlineIntersects[j] is near then iP then
                iP := outlineIntersects [j]
            endif
        end if
        j := j+1
    end while
    return true
[P2] :   else
    // don't intersect ray with outline object
    while slabIntersects[] is not empty do
        if slabIntersects[i] is near than iP then
            iP := slabIntersects[i]
        end if
        i := i+1
    end while
    return true
    end if
[P3] :   else
    // don't intersect ray with slab object
    if TraversalSubKDTree(CSG::subtree, R, outlineIntersects[],
IObject) then

```

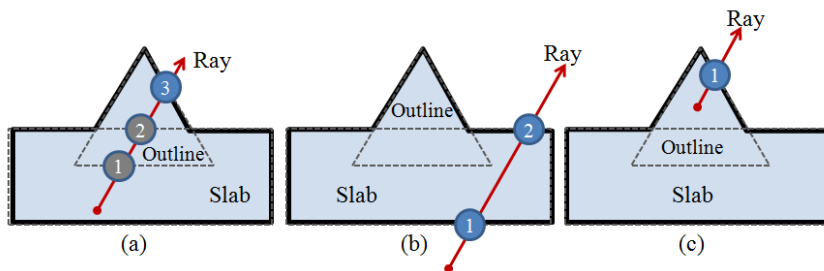
```

// intersect ray with outline object
while outlineIntersects[] is not empty do
    if outlineIntersects[j] is near then iP then
        iP := outlineIntersects [j]
    endif
    j := j+1
end while
return true
else
    iP := NULL
    return false
end if
end if
else
    return Traversal of conventional CSG Union object
end if
end function

```

---

Algorithm 6 is a new traversal algorithm for the Slab-Outline CSG union object. The algorithm has somehow complicated logic, so I will explain this with simple intersection cases. Figure 18 describes these simple three intersection cases.



**Figure 18.** Three cases of intersection between a ray and a Slab-Outline CSG union object.

Algorithm 6-[P1] processes Figure 18(a). In this case, the ray meets with

the Slab-Outline CSG union object at three intersection points marked with number 1,2, and 3. The number 2 is an intersection point with the slab object, and number 1 and 3 are intersection points with a single object of pyramidal texture outlines. Algorithm 6-[P1] tests all intersection points, whether each intersection point exists in other objects or not. If a point exists inside other object's volume (or area in 2-D), then the point is removed from a valid intersection point set. Thus, Figure 18(a) case has only one valid intersection point (*i.e.*, Number 3) since Number 1 exists inside the slab, and Number 2 exists inside the outline.

Algorithm 6-[P2] processes Figure 18(b) in which the ray meets only the slab object at two intersection points (*i.e.*, Number 1 and 2). Thus, it does not need to check inside because the ray does not meet any outline objects. Algorithm 6 will, therefore, return Number 1 as the nearest intersection point.

Algorithm 6-[P3] processes Figure 18(c). At that time, the ray meets only one of the outline objects at one intersection point, Number 1. Also, inside checking process is not needed because the ray does not meet the slab object. Number 1 is a valid intersection point and also the nearest intersection point, as a result, Algorithm 6 returns Number 1 as an intersection point.

---

**Algorithm 7.** Traversal of Slab-Outline CSG Difference object

---

```
function TraversalCSGDifference(Object CSG, Ray R, Point &iP) return
boolean

if CSG is slab-outline then
    if TraversalPrimitive(CSG::elements[0], R, slabIntersects[]) then
        // intersect ray with slab object
[P1] : if TraversalSubKDTTree(CSG::subtree, R, outlineIntersects[], IObject)
then
    // intersect ray with outline object
    while slabIntersects[] is not empty do
        if slabIntersects[i] doesn't exist in IObject then
            if slabIntersects[i] is near than iP then
                iP := slabIntersects[i]
```

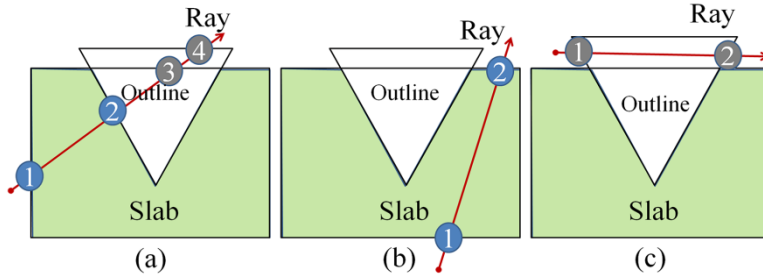
```

        end if
    end if
    i := i+1
end while
while outlineIntersects[] is not empty do
    if outlineIntersects[j] exists in CSG::element[0] then
        if outlineIntersects[j] is near then iP then
            iP := outlineIntersects [j]
        endif
    end if
    j := j+1
end while
return true
[P2] : else
    // don't intersect ray with outline object
    while slabIntersects[] is not empty do
        if slabIntersects[i] is near than iP then
            iP := slabIntersects[i]
        end if
        i := i+1
    end while
    return true
end if
[P3] : else
    // don't intersect ray with slab object
    iP := NULL
    return false
end if
else
    return Traversal of conventional CSG Difference object
end if
end function

```

---

Next, I will explain Algorithm 7 that traverses a Slab-Outline CSG difference object as Figure 19 in which the colored region is a result volume (or area in 2-D) of CSG difference operation.



**Figure 19.** Three cases of intersection between a ray and a Slab-Outline CSG difference object.

Algorithm 7-[P1] processes Figure 19(a) in which the ray meets with the Slab-Outline CSG difference object at four intersection points marked with Number 1,2,3, and 4. Number 1 and 3 are intersection points with the Slab, and Number 2 and 4 are intersection points with a single object of outline objects. Algorithm 7-[P1] tests Number 1,2,3 and 4 whether each intersection points exist inside the volume of other objects or not. If an outline's intersection point does not exist inside the slab, the point is removed from a valid intersection point set. On the other hand, if a slab's intersection point exists inside the outline object, the point is removed from a valid intersection point set. In this case, the inside check logic is different from the logic of Algorithm 6 because the outline object's volume is entirely removed from the slab object and the outline's volume does not contribute at the total CSG difference object. Thus, Figure 19(a) has two valid intersection points (*i.e.*, Number 1, 2) because Number 3 exists inside the outline object, and Number 4 does not exist inside the slab object.

Algorithm 7-[P2] processes Figure 19(b). In this case, the ray meets only the slab object at two intersection points(*i.e.*, Number 1 and 2), and it does not need to check inside because the ray does not meet any outline objects. As a result, Algorithm 7 will return Number 1 point as the nearest intersection

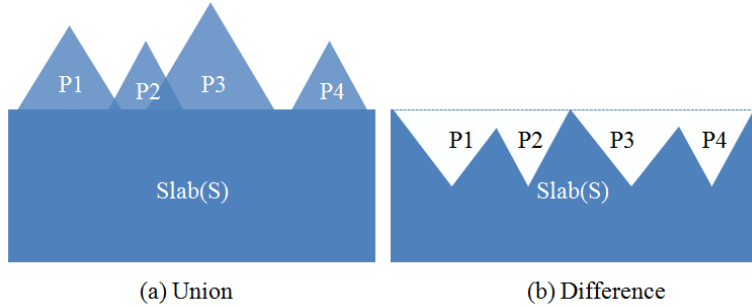
point.

Next, Algorithm 7-[P3] processes Figure 19(c) in which the ray does not meet the slab object. Therefore, the traversal function of the sub kd-tree does not need to be called because, like Number 1 and 2 in Figure 19(c), any of the intersection points will not exist inside the slab object. Thus, Algorithm 7 will return false and a null intersection point.

Until now, The Slab-Outline algorithm was introduced for the non-overlapped pyramidal texture silicon solar cells. However, the non-overlapped case is a special case of the randomly textured silicon solar cells. Thus, a more general algorithm is needed to analyze the randomly textured silicon solar cells.

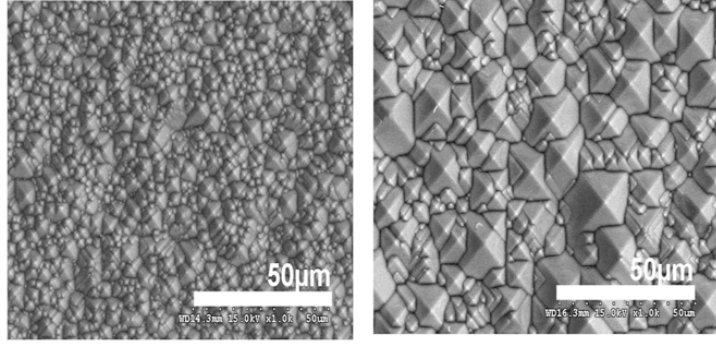
### 2.2.2 Overlapped pyramidal texture case

Figure 20 shows 2-D sliced silicon solar cells, which have randomly distributed pyramidal patterns. All or some of the pyramidal patterns may be overlapped with each other, and all these patterns have overlapped region with the substrate slab. Figure 21 shows real surface images of the commonly used textured silicon solar cells.



**Figure 20.** Schema for overlapped texture of silicon solar cells. (a) is normal pyramidal shape, and (b) is inverse pyramidal shape.





**Figure 21.** SEM images of pyramidal texture of silicon solar cells.

The two objects of Figure 20 also can be made by CSG union and difference operations. Such that Figure 20 (a) is a CSG union and (b) is a CSG difference, respectively. However, in this case, all or some of the pyramidal primitives may participate in CSG operation with each other and with the substrate slab. So, Equation (2.3)-(2.5) have to be modified as follow :

$$\{ S \cup \{P_1, P_2, P_3 \dots, P_{n-1}\} \} = (a) , \quad (2.13)$$

$$\{ S - \{P_1, P_2, P_3 \dots, P_{n-1}\} \} = (b) , \quad (2.14)$$

$$\{P_i \cap P_j\} \neq \emptyset, \exists(i \neq j). \quad (2.15)$$

Equation(2.13) and (2.14) are same as Equation(2.3) and (2.4), respectively. However, Equation(2.15) is different from Equation(2.5). These facts reveal that all previous algorithms and structures for the non-overlapped texture case can be used except for Algorithm 5-7.

To solve traversal problems for the Slab-Outline CSG objects and the sub kd-tree, which contain the randomly textured silicon solar cells, Algorithm 5-7 was modified like these

---

**Algorithm 8.** Traversal Sub KD-tree for overlapped texture

---

**function** TraversalSubKDTree(KD-tree root N, Ray R, Point[] iP, *Object[]* *IObjects*) **return** boolean

(I\_min, I\_max) := intersect R with N's bounding box

**if** ray does not intersect bounding box **then**

iP[] := empty; IObjects[] := empty; **return** false

```

end if
boolean bIntersect := false
push ( N, I_min, I_max) to stack
while stack is not empty do
    (newNode, newI_min, newI_max) := pop stack
    while newNode is not a leaf do
        tplane := (newNode split position[axis] -
        R::Initial[axis])/R::Direction[axis]
        if tplane <= newI_min then
            newNode := right child node; continue
        end if
        if tplane >= newI_max then
            newNode := left child node; continue
        end if
        push (right child node, tplane, newI_max) to stack
        newNode := left child node
        newI_max = tplane
    end while
    if newNode is not empty then
        while newNode::primitive[i] is not NULL do
            Point[] currentPointList := empty
            if ray intersect with newNode::primitive[i] then
                bIntersect := true
                iPoint[] = intersection points
[P1]:    while iPoint[j] is not NULL do
                    boolean bInside := false
                    while IObject[k] is not NULL do
                        if iPoint[j] exist in IObject[k] then
                            bInside := true; break
                        end if
                        k := k + 1
                    end while

```

```

        if bInside is false then
            insert iPoint[j] at currentPointList[]
        end if
        j := j + 1
    end while
    Point[] tempPointList
[P2]:   while iP[j] is not NULL do
        if iP[j] does not exist in newNode::primitive[i] then
            tempPointList[k] := iP[j]; k := k + 1
        end if
        j := j + 1
    end while
    iP[0] := Nearest (tempPointList[] + currentPointList[])
    iP[1] := Farthest (tempPointList[] + currentPointList[])
    IObject[] := IObject[] + newNode::primitive[i]
    end if
    end while
    end if
    end while
    return bIntersect
end function

```

---



---

**Algorithm 9.** Traversal of Slab-Outline CSG Union object

---

**function** TraversalCSGUnion (Object CSG, Ray R, Point &iP) **return** boolean

```

    if CSG is slab-outline then
        if TraversalPrimitive(CSG::elements[0], R, slabIntersects[]) then
            // intersect ray with slab object
            if TraversalSubKDTTree(CSG::subtree, R, outlineIntersects[], IObject[])
                then
                    // intersect ray with outline object
                    while slabIntersects[] is not empty do

```

```

    boolean bInside := false
    while IObject[] is not empty do
        if slabIntersects[i] exist in IObject[j] then
            bInside := true; break
        end if
        j := j + 1
    end while
    if bInside is false and slabIntersects[i] is near than iP then
        iP := slabIntersects[i]
    end if
    i := i+1
end while

```

[ ... ] : Same as Algorithm 5.

**end function**

---



---

**Algorithm 10.** Traversal of Slab-Outline CSG Difference object

---

```

function TraversalCSGDifference(Object CSG, Ray R, Point &iP) return
boolean
    if CSG is slab-outline then
        if TraversalPrimitive(CSG::elements[0], R, slabIntersects[]) then
            // intersect ray with slab object
            if TraversalSubKDTree(CSG::subtree, R, outlineIntersects[], IObject[])
            then
                // intersect ray with outline object
                while slabIntersects[] is not empty do
                    boolean bInside := false
                    while IObject[] is not empty do
                        if slabIntersects[i] exist in IObject[j] then
                            bInside := true; break
                        end if
                        j := j + 1
                    end while

```

```

    if bInside is false and slabIntersects[i] is near than iP then
        iP := slabIntersects[i]
    end if
    i := i+1
end while
[ ... ] : Same as Algorithm 6.
end function

```

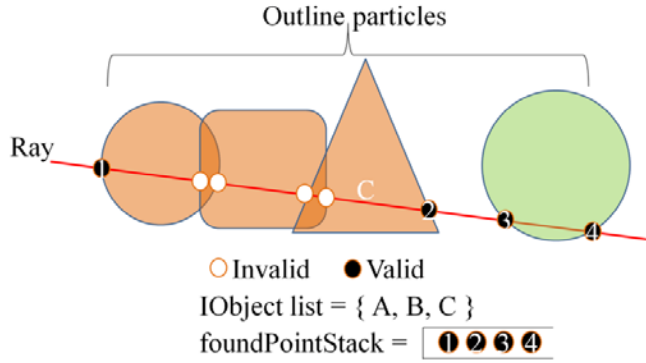
---

Algorithm 8 for the sub kd-tree is different from the conventional kd-tree traveling algorithm and from Algorithm 5. As described in the previous section, the conventional kd-tree algorithm only returns the single nearest intersection point without intersected object, so this cannot be used to traverse a CSG object or overlapped objects. To solve this problem, Algorithm 5 was proposed. However, this solved only CSG operations for the non-overlapped outlines and the slab substrate with the multi nearest intersection points and the single intersected object. Thus, to solve the overlapped outlines problem, Algorithm 8-[P1] and [P2] are designed for the multi intersection points and the multi intersected objects.

Algorithm 8-[P1] checks, whether new intersection points exist inside the volume of the previously saved IObject's elements or not. If the newly founded intersection point does not exist in all objects of the IObject list, the point is saved in the currentPointList, which is used to store valid newly founded intersection points. Algorithm 8-[P2] checks, whether points of the iP list exist in the current object or not. If a point of the iP list does not exist in the current object, the point is saved in the tempPointList which is used to store valid points of the iP list.

As a result, Algorithm 8 saves the filtered points among the currentPointList and the tempPointList at the iP list. In addition, it saves the current object at the IObject list. The iP list and the IObject list are returned parameters of Algorithm 8. Therefore, the information of the overlapped outline objects and the intersection points can be achieved, which indicate where the ray intersects with the overlapped outline object's outer boundary as Fig-

ure 22. With these information and Algorithm 9 - 10, the overlapped texture problem can be solved.



**Figure 22.** Result of the **Algorithm 8**.

Through this section, I introduced the Slab-Outline algorithm, which makes possible to analyze numerically the effects of the enhanced efficiency of the arbitrarily textured silicon solar cells. In the following section, the Slab-Outline algorithm was validated with showing exactness of backward tracing images and comparing the simulated reflectance data of the arbitrarily textured silicon solar cell to the measured data. Moreover, I checked time complexity of the algorithm and showed error analysis result briefly.

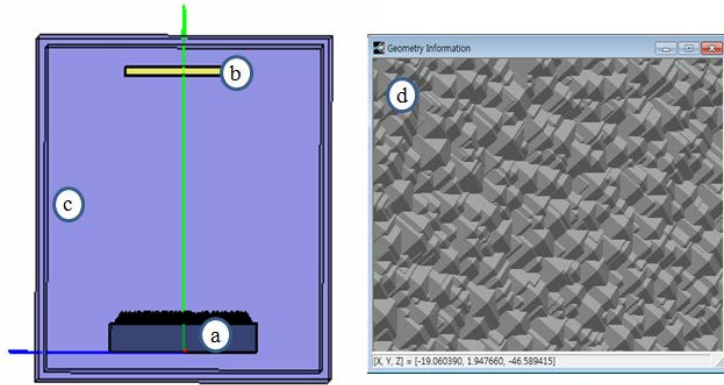
## 2.3 Numerical Results : Validation

### 2.3.1 Examine of Backward Ray Tracing Results

At first, a backward ray tracing method was used to validate the Slab-Outline algorithm. The backward ray tracing was used to make photorealistic images in computer graphics so the validity of the Slab-Outline algorithm can be easily checked with their resulting scenes. If the Slab-Outline algorithm is incorrect, the output image will show incorrect scene. The backward ray tracing system is composed of targeting system geometry, light and camera. For this simulation, 'RayWiz-SOLAR' [27] which include the Slab-Outline algorithm and a photon map [4] was used.

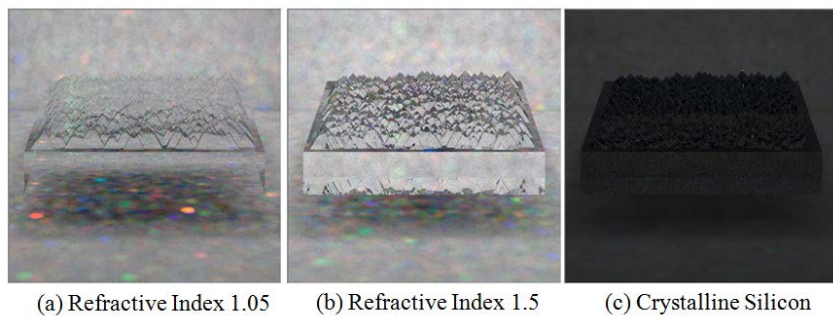
The simulation model had one substrate silicon slab, which merged 750

randomly positioned pyramidal objects, one box shaped light, and one chamber which has five Lambert BSDF walls as Figure 23. The light source had Standard-E spectrum. The camera was located at position (600, 0, 0) and look at direction(-1, 0, 0) and had angle of 15°.



**Figure 23.** Geometry of backward ray tracing system. (a) is textured silicon, (b) is box light, (c) is a chamber, and (d) is top surface geometry of the (a).

Three types of material properties for the randomly textured silicon were prepared. The first property of the material is lossless and fictitious refractive index value 1.05. This value was chosen to show a non-refracted shape of the textured silicon. The second is lossless and refractive index value 1.5, which is similar to that of glass. The third is measured refractive index and absorption coefficient data of crystalline silicon, which is broadly used to make silicon type solar cell, and this data set was reported in Appendix I.



**Figure 24.** Simulation results of the backward ray tracing.

Figure 24 shows the results of the backward ray tracing. As you can see, the overlapped pyramidal shape particles were exactly simulated by the Slab-Outline algorithm through all cases. If you compare Figure 24(a) with (b), then you can find two kinds of differences. One is a difference of transmittance, such that, Figure 24(a) is more translucent than Figure 24(b). If a refractive index 1.0 is used to simulate Figure 24(a), the textured silicon would be disappeared from Figure 24(a) because the refractive index 1.0 is same as free space's value. The other difference is their reflectance value. The textured silicon of Figure 24(a) does not show any reflection phenomena except for a little surface reflection. However, the textured silicon of Figure 24(b) shows reflected texture shapes on its bottom region and stronger surface reflection than Figure 24(a).

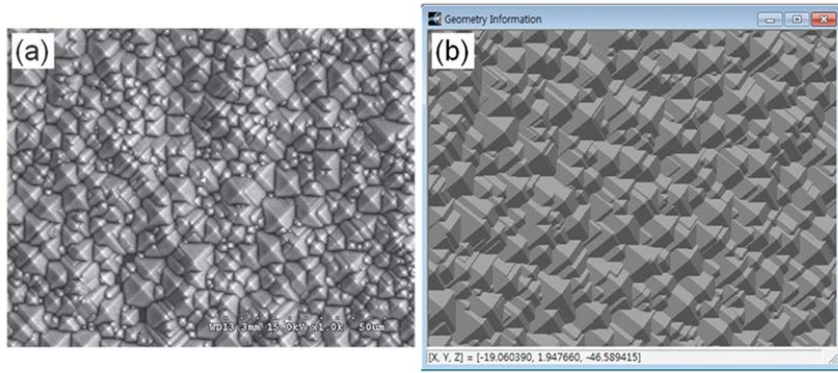
Figure 24(c) shows a significant result. Figure 24(a) and (b) have zero absorption coefficient, so all rays, which traveled through the textured silicon of two systems, do not experience any energy losses. However, Figure 24(c) has an absorption coefficient in a given spectrum region, so all rays' energies decrease according to the Lambert-Beer's law. As a result, Figure 24(c) is darker than Figure 24(a) and (b).

These backward results showed the exact shape of the pyramidal texture and the expected optical phenomena. Thus, these results indicate the correctness of the Slab-Outline algorithm with intuitive and qualitative method.

### **2.3.2 Comparison of Experimental Results**

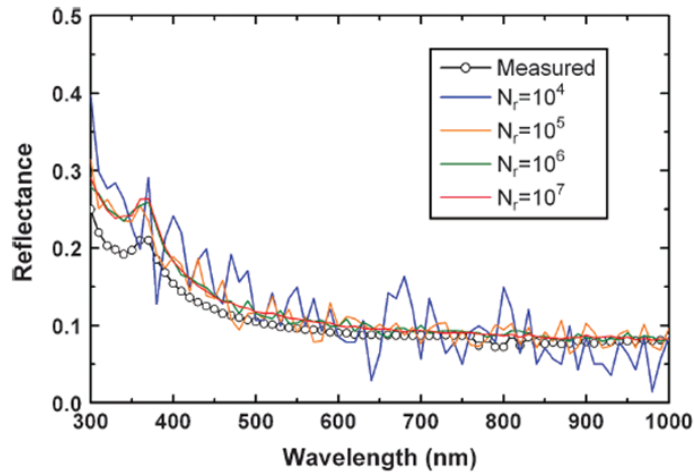
In this section, the numerical analysis results simulated with the Slab-Outline algorithm were compared to the measured reflectance values to verify the validation of the algorithm. For this experiment, a silicon wafer, which has 250 $\mu\text{m}$  thick substrate and textured pyramidal patterns of an average size of 10 $\mu\text{m}$  using KOH/IPA solution [12], was prepared. The surface morphology of the textured Si sample and a 3-D rendering image of the geometric information for the simulation is shown in Figure 25(a) and (b), respectively [12].





**Figure 25.** (a) Surface morphology of a textured Si sample and (b) a 3-D rendering image of the geometric information for the simulation [12]

The total reflectance of the Si sample was measured in the 300 - 1000nm ranges at a normal incidence angle. During the simulation, the energy of each reflected ray was collected by an energy detector positioned on the top of the Si surface and with the same surface dimension as that of the Si system [12]. The total optical energy reflectance was obtained by summing up the individual energies of all the reflected rays of a predetermined number [29].



**Figure 26.** Comparison of the measured and simulated reflectance spectra [12].

In Figure 26, the simulated reflectance spectra for  $N_r$  values of  $10^4$ ,  $10^5$ ,  $10^6$ , and  $10^7$  rays were compared with the measured reflectance from the pre-

pared Si sample. For the  $N_r = 10^4$ , the simulated spectrum showed a large amount of fluctuations and could not reproduce the measured spectrum properly because the value of  $N_r$  was too small when compared to the system size. As  $N_r$  increased to  $10^5$  and  $10^6$  rays, the fluctuation in the simulated spectrum became smaller, showing better agreement with the overall trend of the measured spectrum to some extent. At  $N_r = 10^7$  rays, the overall shape of the simulated spectrum corresponded the measured spectrum very well [12]. With this result, I validated the Slab-Outline algorithm in a quantitative way.

### 2.3.3 Error Analysis

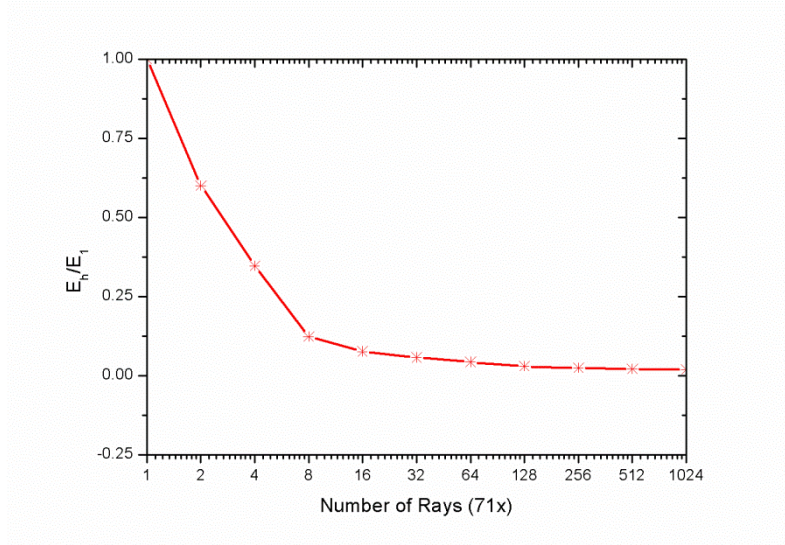
The previous model was simulated with eleven kinds of total number of rays, such that 71x (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024). The common multiplier '71' came from the number of spectrums, which were used for spectral analysis of reflectance in the system. In this case, the exact solution of the system could not be used, so I used the measured data which was used in the previous section. The error value was calculated according to the equation

$$E_h = h \sum_{i=1}^S |U - \tilde{U}|_i, \quad (2.16)$$

where  $E_h$  is Error,  $h$  is  $\frac{1}{N_r}$ ,  $S$  is number of spectrum,  $U$  is measured (exact) value, and  $\tilde{U}$  = simulated value. The results are like **Table 1** and Figure 27.

$N_r$	71	142	284	568	1136	2272	4544	9088	18176	36352	72704
$h$	1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512	1/1024
$E_h$	0.89338	0.53630	0.30944	0.10967	0.06808	0.05107	0.03827	0.02618	0.02168	0.01882	0.01678
$E_h/E_1$	1.00000	0.60030	0.34637	0.12276	0.07620	0.05716	0.04283	0.02931	0.02427	0.02107	0.01878

Table 1. Error Analysis of Slab-Outline Algorithm.



**Figure 27.** Relative errors according to the increased number of rays.

To calculate order of convergence of this new ray tracing algorithm, which includes the Slab-Outline algorithm, Table 1 and the Least Square Method (LSM) are used. Equation (2.17) shows the calculated order of convergence.

$$O(h) = Ch^\alpha = 31.34369 * h^{0.83143}, \quad (2.17)$$

where  $C$  is 31.34369 and  $\alpha$  is 0.83143.

As Table 1 shows, if the number of rays are increased, the simulation result converges to the measured values (exact solution). This is another evidence of the correctness of the Slab-Outline algorithm.

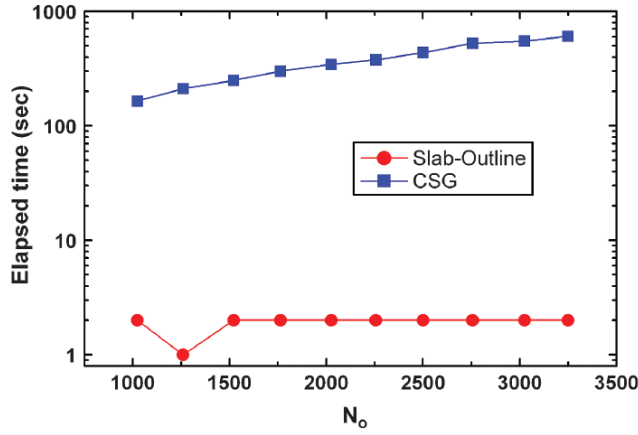
### 2.3.4 Time Complexity

The time complexity of the Slab-Outline algorithm was analyzed by the conventional traversal algorithm of the kd-tree. This has  $O(\log N_o)$  average time complexity [28, 43]. As you can see from Algorithm 5, the non-overlapped Slap-Outline algorithm has almost same logic of the conventional algorithm. Therefore, in this case, the average time complexity of the Slap-Outline algorithm is also  $O(\log N_o)$ . However, Algorithm 8 (overlapped case) has additional check logic, so its average time complexity need to be analyzed more carefully.

Algorithm 8-[P1] and [P2] are these extra check logic. They check where

or not the previously found intersection points exist in a newly found object, and also the newly found intersection points exist in the previously found object list. In this case, the time complexity is mainly affected by the number of the previously found objects. This number may be thought of as 'm' of [43]. In the paper,  $S_m(X_q)$ , which was defined by Equation(1) at [43], was assumed sufficiently small compared to the total number of objects ( $N_o$ ), so 'm' is also sufficiently small. Also, the inside check time is same as the intersection search time. With these two facts, that the average time complexity of the Slap-Outline is also  $O(\log N_o)$  can be easily proved.

For more intuitive comparison, the efficiency of the *Slab-Outline* algorithm was tested by comparing the computing time between the *Slab-Outline* algorithm and the CSG algorithm [12]. The CSG algorithm was chosen because this gives exact solution for a ray tracing of merged objects. In the comparative simulation,  $N_r$  was fixed at 105 rays, and the variation of the computing time was monitored while increasing the number of pyramidal patterns ( $N_o$ ) of a regular array from 1024 to 3249 in 256 steps. The monitored computing time in both algorithms is plotted in Figure 28. It is clear that the computing time of the *Slab-Outline* algorithm does not change regardless of any increase in  $N_o$ , remaining in the 1–2 sec range. On the other hand, the computing time of the CSG algorithm increased monotonically as  $N_o$  increased. Even with the smallest value of  $N_o$  of 1024, the *Slab-Outline* algorithm provided a computing speed that was more than 80 times faster than that by the CSG algorithm, revealing the superiority of the efficiency of its logic [12].



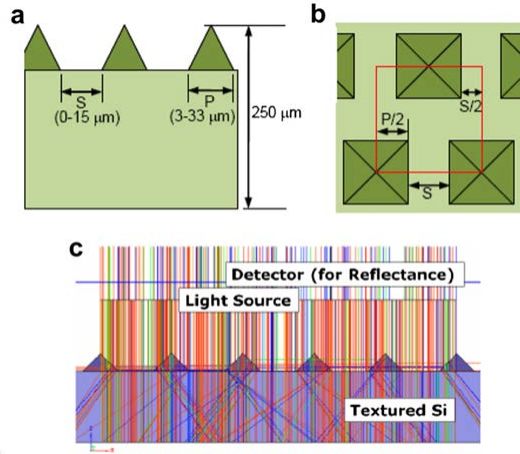
**Figure 28.** Performance comparison of Slab-Outline against CSG algorithm [12].

## 2.4 Numerical Analysis : Applications

As an application of the Slab-Outline algorithm, the size and density of a pyramidal texture, as formed on the surface of Si solar cells to increase the light trapping efficiency, was examined on the optical reflectance and absorptance with comparing simulation results and experimental observations [44].

### 2.4.1 Simulation

Schematics of a cross-sectional and plan view for the simulation geometry and dimension are depicted in Figure 29(a) and (b), respectively. The thickness of the Si substrate was set to 250 $\mu\text{m}$ . In one simulation, the pitch of the unit pyramid varied from 3 to 33 $\mu\text{m}$  in 2 $\mu\text{m}$  steps. Meanwhile, the space between the pyramids set to zero such that the entire surface was covered with a pyramidal pattern. In the other simulation, the pitch of the unit pyramid was fixed at 15 $\mu\text{m}$  and the space varied from 0 to 15 $\mu\text{m}$  in 1 $\mu\text{m}$  steps, i.e., varying the density of the pyramid. It was assumed that the distance between the substrate bottom and the vertex of the pyramid remained the same as the substrate thickness [44].

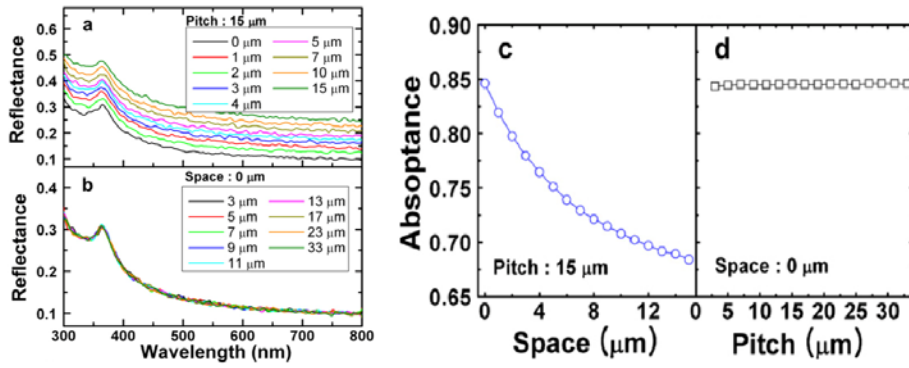


**Figure 29.** Schematics of the cross-section (a) and plan view (b) for the simulation geometry, and (c) optical simulation structure for the reflectance and absorption [44].

Optical simulation was carried out using the 'Raywiz-SOLAR' [27]. Total input light energy of 100 W in the wavelength range 300-800 nm was used. As depicted in Figure 29(c), the light impinges on the textured surface at normal incidence; the absorbed energy is directly calculated inside the Si medium while the reflected energy is gathered at a detector positioned at the top of substrate [44]. The direct calculation algorithm of absorption energy can be found in Chapter 3 and the literature [29].

#### 2.4.2 Results and discussion

Figure 30(a) and (b) are the reflectance spectra as the space varies with a fixed pitch of 15 μm as well as the reflectance spectra with a varying pitch with no space (*i.e.*,  $S = 0 \mu\text{m}$ ), respectively. The corresponding variations in absorbance averaged in the 300-800 nm ranges are shown in Figure 30(c) and (d). With increasing space between the pyramids at a fixed pitch, the reflectance increased gradually, resulting in a monotonic decrease in the absorbance (Figure 30(a) and (c)). On the other hand, the reflectance spectra did not show any dependence on the pitch, indicating that the size of the pyramid does not affect the absorbance at all in the examined range (Figure 30(b) and (d)) [44].

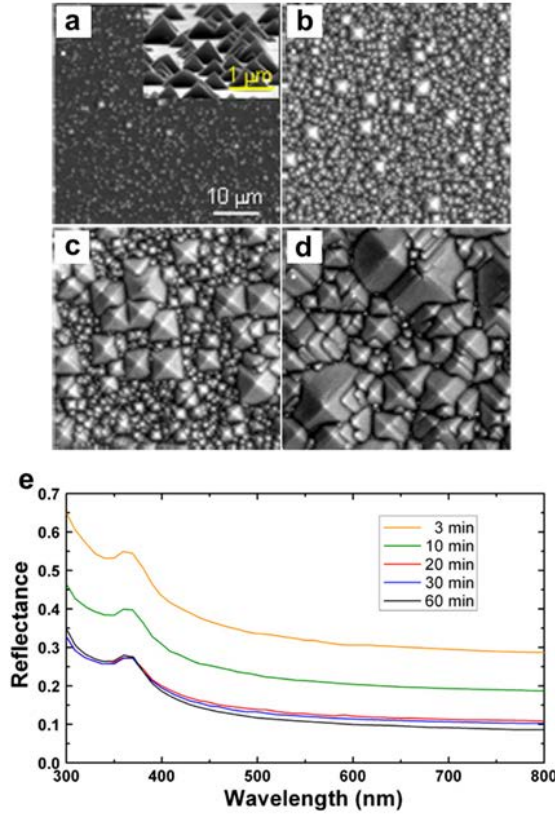


**Figure 30.** Simulated reflectance spectra of textured surface with various spaces (a) and pyramid sizes(b). (c) and (d) are the average absorptance corresponding to (a) and (d), respectively [44].

To compare the simulated dependence of the optical properties on the size and the space, a 250μm thick Si (100) wafer was etched in a KOH-Isopropanol solution while varying the etching time. SEM images (Figure 31(a)-(d)) of the textured surfaces and the corresponding optical reflectance spectra (Figure 31(e)) were then obtained. A SEM micrograph of a sample etched for 3 min (not shown here) exhibited only small hill-like humps without the formation of pyramids. After 10 min of etching, a pyramid less than 1μm in size were formed with a relatively large space between them, and a large drop in the reflectance was observed. The surface etched for 20 min showed a mixed texture of somewhat large pyramids 3-4μm in size and smaller ones with a size of less than 1μm. A somewhat small space region could be seen, and the reflectance was lowered substantially [44].

After 30 min of etching, nearly the entire surface was covered with pyramids 6-8μm in size together with a large number of smaller ones. With a prolonged etching time of 60 min, the pyramid size continued to increase, showing pyramids as large as 10-15μm. It is notable that the difference in the reflectance spectra of the samples etched for 20 and 30 min is marginal and that even the reflectance spectra of the sample etched for 60 min did not show a significant reduction compared to those of the samples etched for 20 and 30 min despite the large difference in the pyramid size distribution [44]. Similar behavior of the reflectance spectra with the textured pyramid size was also

observed in a previous study [45].



**Figure 31.** SEM micrographs of Si surfaces etched for (a) 10, (b) 20, (c) 30 and (d) 60 min. (e) The measured reflectance spectra of the etched Si surfaces [44].

The simulated and the experimentally measured results clearly show the size-independent behavior of the optical properties. The slight lowering of the measured reflectance spectra with the increased pyramid size shown in the samples etched for 30 and 60 min can be attributed to the decreased probability of rendering the relatively flat area that would be present in the valleys between the pyramids, as was observed previously [21]. These results imply that it is important to fabricate pyramid patterns with sharp valleys, edges and peaks with no flat space so as to maximize light trapping in Si solar cells [44].



## 2.5 Conclusion

In this chapter, I proposed an novel and robust collision detection algorithm so called the Slab-Outline algorithm. It can simulate the absorption energy of the arbitrarily textured Si solar cells with superior efficiency. The validity of the Slab-Outline algorithm was shown with backward ray tracing results and with comparing the simulated reflectance spectra with the measured spectra from an arbitrarily textured Si surface.

The efficiency of the computing time was proven by comparing the computing time between the Slab-Outline and the CSG algorithm known to offer the exact solution of ray tracing with merged objects.

Within my knowledge, the Slab-Outline algorithm is the first algorithm which can precisely intersect the 3-D randomly textured thin film solar cells with  $O(\log N)$  complexity.

The Slab-Outline algorithm also can be used to synthesize photorealistic images, simulate efficiency of Back Light Units(BLU), and analyze any other simulation parts, especially when an optical system that has a lot of surface particles is wanted to simulate.

## 3 Simulation with Ray Tracing Method

### 3.1 Overview

In this chapter, I propose an efficient ray tracing algorithm to implement the Lambert-Beer's law and  $2 \times 2$  characteristic matrix method for analyzing absorption energy of the thin film silicon solar cells directly. In the previous optical simulation researches, the estimation of light absorption in an individual layer was limited to the case of totally flat surface and interfaces, and usual simulation steps were composed of calculating the reflectance of individual layer and determining the energy absorbed by combining the absorption coefficient and the incoming light energy, which was determined from the calculated reflectance [29]. In other words, previous simulation utilized a coherent  $2 \times 2$  characteristic matrix method [81], combined with a weighting factor from the characteristics of bidirectional scattering distribution function (BSDF) [79,80].

Unfortunately, this kind of approximated method cannot guarantee the accuracy in the case where random optical paths, which will remove the interference effect, are generated by rough surfaces or other optical geometry distorting light path [29]. Furthermore, the calculated results will provide only relative ratios of absorption inside the component layers, and will have difficulty in exact estimation of absorbed solar energy [29]. Recently, an optical modeling based on ray tracing was proposed [23]. However, this method is only an indirect method, which gave total absorption inside solar cell system by obtaining the total reflectance and transmittance without considering interferences [29].

The newly developed algorithm is based on Monte-Carlo ray tracing techniques, and is capable of calculating the optical absorption energy directly by separating the light passing through the medium into the coherent part and the incoherent part in the course of non-sequential ray tracing throughout the whole region of solar cell structure. Therefore, the algorithm can provide direct calculation of the accurate amount of absorbed energy in the individual

layers regardless of coherent and/or incoherent light inside the medium [29]. This is the first method to simulate the absorption energy of the thin film solar cells having arbitrary surface property with non-sequential ray tracing method in my knowledge.

### 3.2 Algorithm

Each ray emitted from light source is selected randomly, and is allocated with predetermined polarization, phase and energy information. When each ray faces an object in solar cell structure, it is determined whether the region is the one where the interference effect has to be considered or not based on the given optical information of the region. If the ray is in the coherent region so that the phase information is meaningful, the characteristic matrix method [81] is applied while considering polarization. If the ray is in an 'absorption detector' region, the optical absorption energy is calculated using Poynting vector. Then the ray that is assigned with new direction from Snell's law by complex refractive index proceeds to the next ray tracing step [82, 29].

On the other hand, in the case for the ray to meet an incoherent object surface, the incident angle dependent energies of transmitted and reflected parts are calculated by applying the Fresnel equation while considering the polarization as mentioned in Chapter 1 [83]. During this process, if BSDF information in the object's interface is available, the direction and energy of ray is determined according to the given BSDF information instead of using Snell's law and Fresnel equation [29].

After this procedure, the calculation of absorption energy is performed using Lambert–Beer's formula by considering the complex refractive index when the ray goes into the absorption detector (*e.g.*, active layer). Lambert–Beer's formula states that a photon flux density after passing distance  $L$  in a film layer with an absorption coefficient  $\alpha(\lambda)$  is reduced by a factor of  $\exp(-\alpha(\lambda) \cdot L)$ , giving

$$A(L, \lambda) = A^0(0, \lambda) \exp(-\alpha(\lambda) \cdot L) \quad (3.1)$$

where  $A^0(0, \lambda)$  is the incident photon flux density per unit area per unit time and per unit wavelength. It is related to the power density  $P(\lambda)$ , which is associated with the solar radiation by

$$A^0(0, \lambda) = P(\lambda) \frac{\lambda}{hc} \quad (3.2)$$

where  $h$  is Plank's constant and  $c$  is speed of light in free space [29].

This process will be continued non-sequentially until the ray energy is entirely exhausted or cannot meet any object in the system. Once the tracing procedure of individual ray is terminated, the total optical absorption energy in the individual layer is obtained by summing up the individual absorption energies of all the rays of predetermined number [29].

$$A_{\text{total}}(\lambda) = \sum_{n=1}^{\# \text{ of traced rays}} A_n(\lambda) \quad (3.3)$$

To implement this algorithm, I used the Algorithm 3 and Algorithm 5 to detect ray and object intersection efficiently and made a absorption detector structure which contains the absorbed energy values and surface area information.

---

#### Class 4. Class for the absorption detector

---

**Class** AbsorptionDetector

**Object reference** pObject

**float array** AbsorbedValue[# of wavelength]

**float** Area

**end Class**

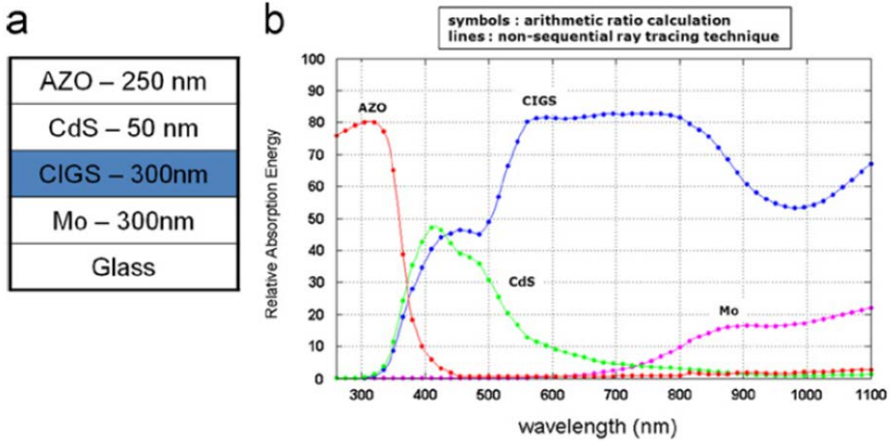
---

where 'pObject' is reference variable for its owner Object class(Class 1) instant, 'AbsorbedValue' is an array type variable for saving the calculated absorption energy according to the wavelength, and 'Area' is a surface area of an owner object that is calculated at the parsing time of the geometry information.

### 3.3 Validation

#### 3.3.1 Case I - coherent system

First, I considered a coherent system in order to validate the correctness of the algorithm by comparing the simulation result obtained by applying the direct calculation algorithm of absorption energy with that obtained using conventional arithmetic ratio calculation. The simulated model is a typical substrate structure comprising AZO(250 nm)/CdS(50 nm)/CIGS(300 nm)/Mo(300 nm)/glass, and comparison of calculated results was made by using standard E spectrum condition with normal incidence while assuming flat surface and interfaces [29].



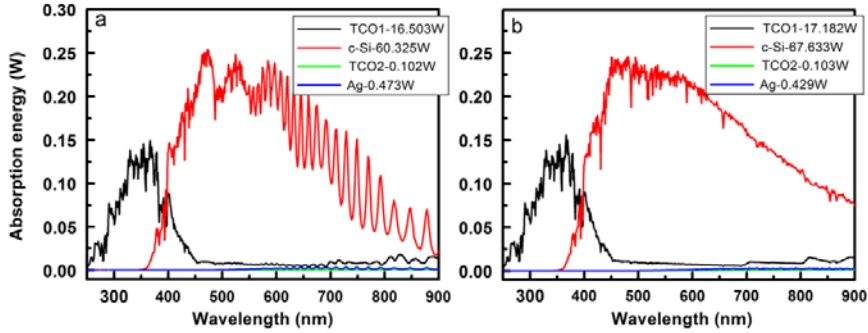
**Figure 32.** (a)Schematics of tested solar cell structure, (b) spectra of absorption ratios determined from arithmetic ratio calculation (symbols) and non-sequential ray tracing technique (lines) [29].

Figure 32 shows the calculated results of relative absorption energies in Mo, CIGS, CdS, and AZO layers obtained from the conventional arithmetic ratio calculation (symbols) and the present method (lines). Clearly, both results correspond very well to each other [29].

### 3.3.2 Case II - incoherent system

In this section, I consider a crystal silicon thin film p-i-n solar cell (c-Si:H).

The simulation system consists of glass(1 mm)/TCO1(1  $\mu\text{m}$ )/c-Si(3  $\mu\text{m}$ )/TCO2(0.1  $\mu\text{m}$ )/silver(1  $\mu\text{m}$ ) in a superstrate structure. Using the same test structure, simulations were carried out for both cases of incoherent condition (with surface roughness) and coherent condition (with flat geometry) [29].



**Figure 33.** Calculated absolute absorption energy spectra for coherent (a) and incoherent (b) conditions [29].

For the calculation of incoherent condition, the angular profiles of the transmitted and reflected ray were assumed to follow Gaussian function type of BSDF. This may be a reasonable assumption because many previous studies have shown that BSDF of textured surfaces had Gaussian function shape [84]. In this simulation, total input light energy of 100W was assumed. The simulated results of the absorption energy in individual layer are compared in Figure 33(a) and (b). The absorption energy spectra obtained for the coherent condition clearly exhibits interference effect. The amounts of absorption energy values in c-Si layers are 60.3 and 67.5W for the structure with flat surface and the structure with textured surface, respectively, indicating more than 11.9% enhancement in absorption for the textured case. The absorption in front TCO also increased from 16.6 to 17.2 W, while those in back TCO and Ag layer were reduced. Total amounts of absorption energy in solar cell structure were 77.4 and 85.35W for the structure with flat surface and the structure with textured surface, respectively. As confirmed from this comparison, in case where

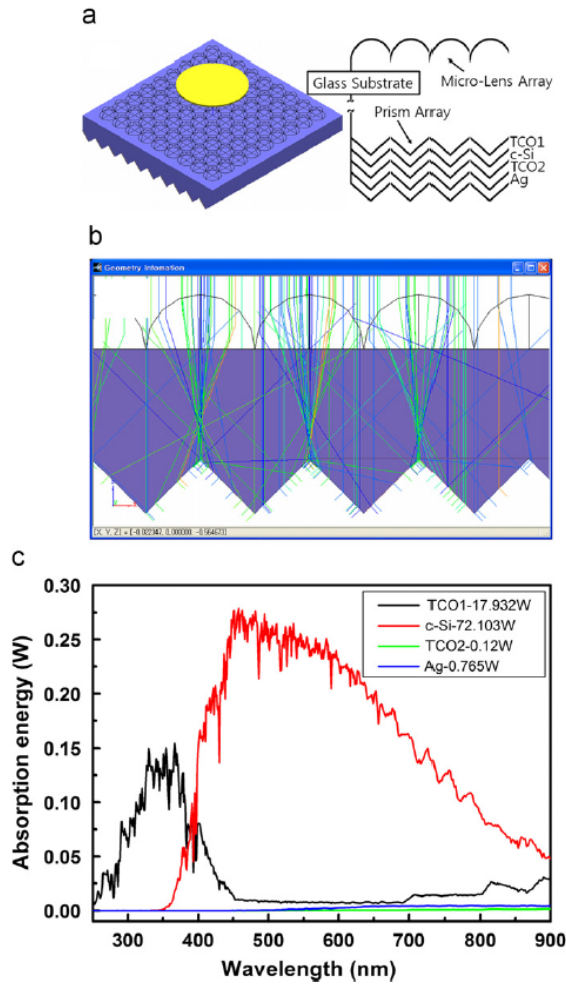
the scattering event in random direction is inherent due to surface roughness, it is necessary to perform the simulation in incoherent condition [29].

### **3.3.3 Case III - coherent + incoherent complex system**

In this section, I consider a solar cell system formed on substrate with complex structure similar to concentrator photovoltaic. As depicted in Figure 34(a), the front and rear sides of glass substrate have assumed the form of 2-dimensional micro-lens array and 2-dimensional prism array with an apex angle of  $90^\circ$ , respectively. And the same cell structure described in Section 3.3.2 is formed on the prism side (Figure 34(a)) [29].

Both the diameter and the length of prism base used in this calculation were 1 mm. Such a large dimension of micro-lens and prism arrays was intended to define the regions for which an incoherent calculation is needed by setting coherent thin film layers and incoherent object region inside system. In this configuration, the individual ray passing through the micro-lens array will diverge in various angles as depicted in Figure 34(b). The ray then proceeds toward the thin film solar cells structure formed on 2-dimensional prism array structure. The transmitted rays will proceed through the solar cell structure with ray path length due to angle variation, and the reflected rays at the prism interface directions move back and forth between the top and the bottom glass interfaces due to large total internal reflection at front glass substrate, leading to enhanced optical path length and subsequently higher absorption inside solar cell structure [29].

The simulation result is shown in Figure 34(c); this configuration reveals much enhanced absorption energy spectra in c-Si layer, especially in the visible range. The amount of absorption energy in c-Si was as high as 72.1 W, which is 19.5% higher than the coherent case and even 6.8% higher than the incoherent case shown in the previous section. The total amount of absorption energy reached up to 90.92W out of 100W of input light energy [29].



**Figure 34.** (a) Schematics and perspective view of simulation system, (b) Example of a ray path inside simulation system, and (c) Calculated absolute absorption energy spectra by combining coherent and incoherent ray conditions [29].



## 3.4 Numerical Analysis : Applications

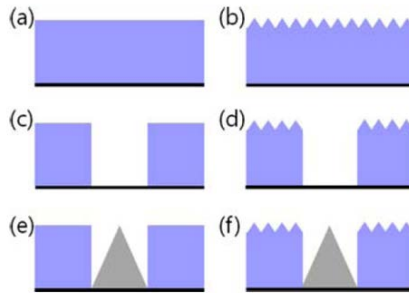
### 3.4.1 High-efficiency Grid-type Si Solar Cell Structure

#### 3.4.1.1 Overview

To find highly efficient solar cell structure, I modeled a new conceptual cell structure that combines two-dimensional grids made of active materials with reflectors in between and reduces the amount of Si usage further. I present a quantitative analysis of the optical efficiency of the proposed grid type cell structure by using the previous algorithm [85].

#### 3.4.1.2 Simulation model

In order to elucidate the quantitative differences in optical efficiencies for various cell structures, I carried out 3-D modeling and relevant optical simulations for the six different cases of hypothetical cell structures shown in Figure 35 [85].

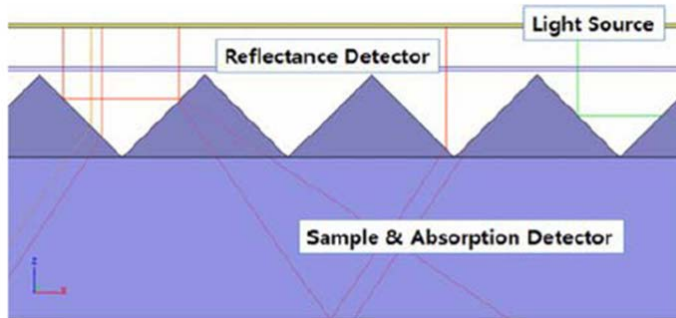


**Figure 35.** Schematics of the six different cell structures examined in simulation [85].

In all cases, the thickness of Si was set to  $200\mu\text{m}$ , and no anti-reflection treatment was assumed. Figure 35(a) depicts the simplest Si solar cell structure with a flat surface, which will lose a considerable amount of incoming light due to the high index of refraction of Si. Figure 35(b) shows a schematic structure with a typical pyramidal surface texturing. In this structure, as shown in Figure 36, part of light reflected from the sidewall of the texture can

be incorporated into Si by a neighboring texture, and the refracted light into Si will have a longer optical path length, thereby enhancing the absorption efficiency [85].

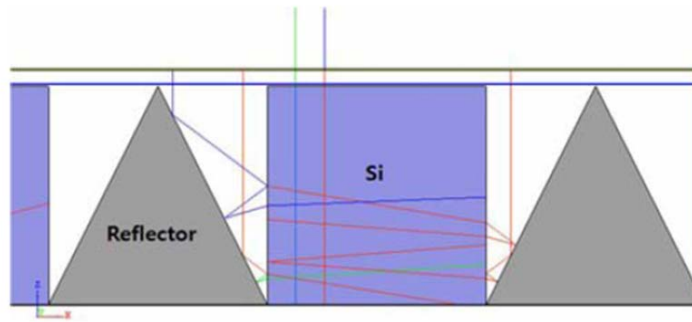
Figure 35(c) shows a simple grid-type Si solar cell structure. The grid pattern was assumed to be two-dimensional lines, and the period of grid pattern was set to  $400\mu\text{m}$ . The structure shown in Figure 35(d) is a combined structure of Figure 35(b) and (c). The schematic shown in Figure 35(e) describes a structure with a reflector structure between the Si grids. The specular reflectance of the reflector surface was assumed to be 92%. In this type of arrangement, as presented in Figure 37, the incoming light falling on the portion of the reflector region is reflected at the reflector surface, is redirected toward the sidewall of the neighboring Si grids, and is then absorbed into the Si. Lastly, Figure 35(f) shows a combined structure of Figure 35(d) and (e). For the simulation of the grid-type cells, the pitch of the grid was set to be one-half the grid period so that the relative volume of Si may be 50% of the cell structure without grids [85].



**Figure 36.** Ray path in Si solar cell with a pyramidal surface texture [86].

The simulation system is composed of 1) the light source plane, which is positioned above the Si surface and generates parallel rays toward the Si surface, 2) the detector plane, which is positioned in the vicinity of the Si surface and collects all the rays reflected from Si, and 3) the sample within which the absorption detector is assumed to be positioned (see Figure 37). The optical simulation was carried out using 'RaywizSOLAR' [27]. The computation was

carried out by using ten million rays for each structure shown in Figure 35 [85].

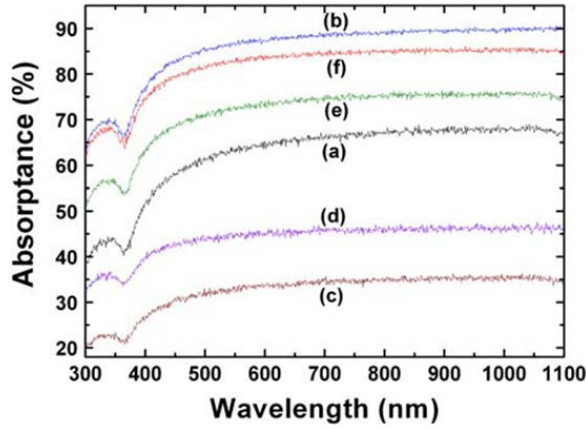


**Figure 37.** Ray path in a grid-type cell structure with reflectors [85].

In these simulations, the light source plane is assumed to be generated a total energy of 100 W in the wavelength range of 300 – 1100nm [85].

#### 3.4.1.3 Results and Discussion

In Figure 38, the absorptance spectra calculated for the six cases of structures depicted in Figure 35 are presented. Structure (b) which is a typical Si solar cell with typical pyramidal texturing had the highest absorptance among all the structures examined, followed by structures (f), (e), (a), (d), and (c) in decreasing order. Notably, for cell structures with flat surfaces, grid-type structure (e) gave better absorption efficiency than structure (a) in spite of the small Si volume. In the case of cell structures with pyramidal texturing, the absorption efficiency of grid-type structure (f) was slightly lower than that of structure (b). However, if the simulation results by taking into account the Si volume is considered, a remarkable difference in optical efficiency per unit Si volume can be found [85].



**Figure 38.** Calculated absorbance spectra of the cell structures shown in **Figure 35** [85].

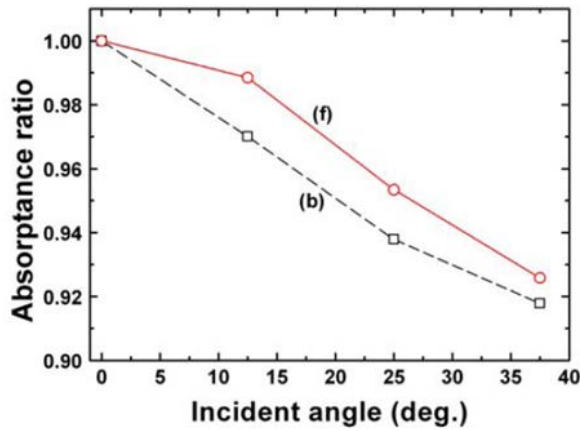
**Table 2** summarizes the relative Si volume used in calculation, the relative absorption energy (*i.e.*, the absorbance with respect to input light energy in the entire spectral range), and the relative absorbance normalized to the Si volume. Structure (b) was used as a reference to calculate the normalized relative absorbance. I note that, even for structure (e) of a simple grid-type without surface texturing, the normalized absorption efficiency increased by as much as 67%. Furthermore, grid-type structure (f) with surface texturing can push the improvement to 91% [85].

Structure	Relative Si volume	Absorbance (%)	Normalized absorbance
(a)	1	62.4	0.73
(b)	1	85.5	1
(c)	0.5	32.5	0.76
(d)	0.5	44.0	1.03
(e)	0.5	71.5	1.67
(f)	0.5	81.7	1.91

**Table 2. List of relative Si volume, absorbance (relative absorption energy) and absorbance normalized by the Si volume [85].**

A closer examination of the spectral response shown in Figure 38 reveals that the grid-type structures with reflectors have an advantage in increasing

the absorption efficiency in the short-wavelength range. Comparing structures (a) and (e), the relative increase in the short-wavelength range is seen to be larger than it is in the long-wavelength range. A similar observation can also be made for structures (b) and (f). That is, the differences in absorptance between two structures in the short-wavelength region are smaller than they are in the long-wavelength region. This stems from increased multiple reflections in the short-wavelength region. This is meaningful in that, for Si solar cells, the reflection loss in the short-wavelength region is much larger than it is in the long-wavelength region because of the high values of the real part of the dielectric function in the short-wavelength region as in Appendix I [85].



**Figure 39.** Comparison of the absorptance ratios of structures (b) and (f) as functions of the incident angle of the incoming light [85].

In order to examine the effectiveness of multiple reflections in the grid and reflector structure, I calculated the dependence of the absorption efficiency on the incident angle of incoming rays. In Figure 39, the absorptance ratios (=absorptance at tilt angle/absorptance at  $90^\circ$ ) of structures (b) and (f) are plotted as functions of the incident angle. From the plot, structure (f) with grids and reflectors clearly exhibits lower rate of decrease of the absorbed energy than structure (b). This results demonstrates that the proposed grid-type structure with reflectors in between provides a practical advantage of enhancing the absorption efficiency by increasing the number of multiple reflections [85].

### **3.4.2 Effect of oxide thin films in back contact on the optical absorption efficiency of thin crystalline Si solar cells**

#### **3.4.2.1 Overview**

For the thin crystalline Si solar cells, maximizing the incorporation of the incoming light into the active layer is of great importance to maintain or even improve the conversion efficiency. One of the main activities for enhancing the absorption energy is texturing and applying antireflection coating for the front surface, and the other is designing the back contact as mentioned in Chapter 1 [17, 86,87]. Proper design of back contact, in which utilized double layered structure composed of thin oxide layer and metal contact, has been shown to be effective in improving the optical absorption efficiency (OAE) in Si by lowering the absorption losses at the metal contact and redirecting the reflected light into Si by total internal reflection (TIR) [88-91]. In order to increase TIR at the c-Si/oxide interface, oxide films with lower refractive index were studied and tested [88]. ZnO based transparent conducting oxide (TCO) thin films have been studied extensively as an alternative for Sn-doped  $\text{In}_2\text{O}_3$  [92-94], and frequently adopted as an oxide layer in back contact. Such transparent oxide films with reasonable amount free carriers inevitably possess non-zero extinction coefficient in long wavelength region of visible and near infra-red region due to free electron absorption [95,96]. However, very little attention has been given to the effect of absorption characteristics of the oxide layer on the OAE in Si based solar cells. Furthermore, analytical studies on the effect of the thickness of thin oxide layer on OAE in Si solar cells are rare [99].

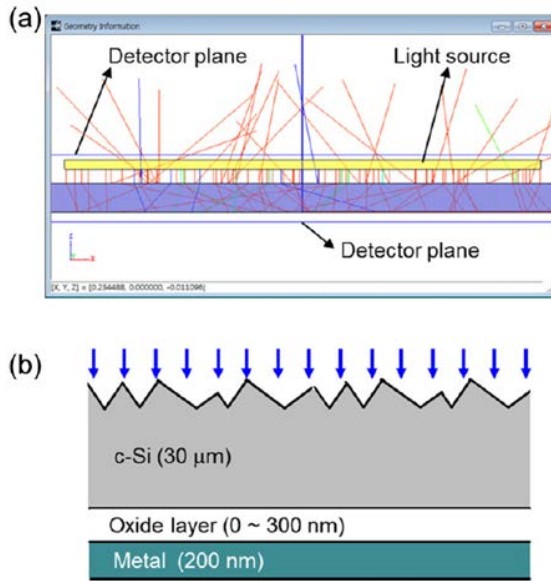
In this study, I examined the thickness dependent OAE in thin c-Si solar cells for various ZnO thin films with different dielectric properties by performing simulations using the ray tracing method. The optical absorption in c-Si and the absorption loss in metal and oxide layer together with the reflection loss were calculated and analyzed for the double-layered back contacts with

undoped ZnO films having different refractive indices and doped ZnO films having different extinction coefficients [99].

#### 3.4.2.2 Simulation model

Figure 40(a) shows the simulation system composed of 1) the light source plane which is positioned above the Si surface and generates parallel rays toward the Si surface, 2) the detector plane which is positioned in the vicinity of Si surface and collects all the rays reflected from c-Si solar cell structure, and 3) the c-Si solar cell structure within which absorption detector are assumed to be positioned. The calculation of absorption energy is performed using Equation(3.1) - (3.3) [99].

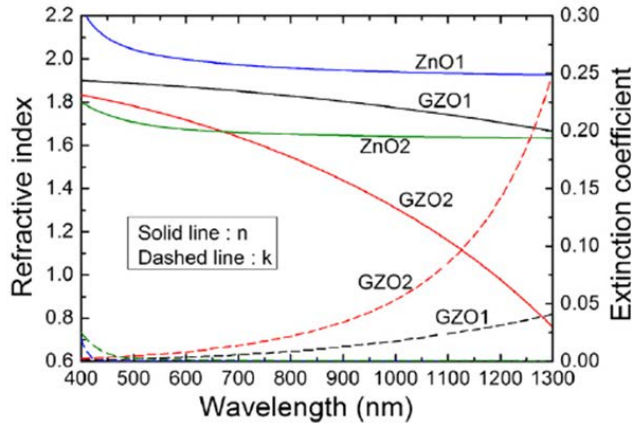
Figure 40(b) depicts the schematic structure composed of c-Si and back contact made of thin oxide layer and metal layer. In all simulations, the thickness of c-Si was set to 30 $\mu$ m. The front and back surfaces of c-Si were assumed to be Lambertian and flat, respectively. Aluminum or silver layer with thickness of 200nm was used as the metal contact. The optical constants of c-Si and metal contact read from optical data base were utilized in the simulations [97]. As oxide layers, two undoped ZnO thin films, which are not conducting and absorptive in visible and near infra-red region, were fabricated by normal magnetron sputtering (ZnO1) and oblique incidence sputtering (ZnO2). Also, Ga-doped ZnO (GZO) films with low ( $1.7 \times 10^{20} \text{ cm}^{-3}$ , GZO1) and high ( $8.6 \times 10^{20} \text{ cm}^{-3}$ , GZO2) carrier concentrations were chosen from the previous study [93], and utilized in numerical analysis as absorptive oxide layers [99].



**Figure 40.** (a) Example of ray paths inside the simulation system, and (b) the schematics of simulated structure [99].

The optical constants of the oxide layer were analyzed by using a simple Drude model combined with the Lorentz oscillator model for transmittance and reflectance spectra [98]. Plots for the optical constants of oxide films are shown in Figure 41. The choice of ZnO1 and ZnO2, which have no absorption in the wavelength higher than 600nm but give significant difference in refractive indices, was made in order to examine the effect of TIR at the c-Si/oxide interface. GZO1 and GZO2 were intended to see the effect of absorption in oxide films. Furthermore, simulations using an artificial oxide film (GZO3) with refractive index of GZO2 but with zero extinction coefficient at all wavelength were also carried out [99].





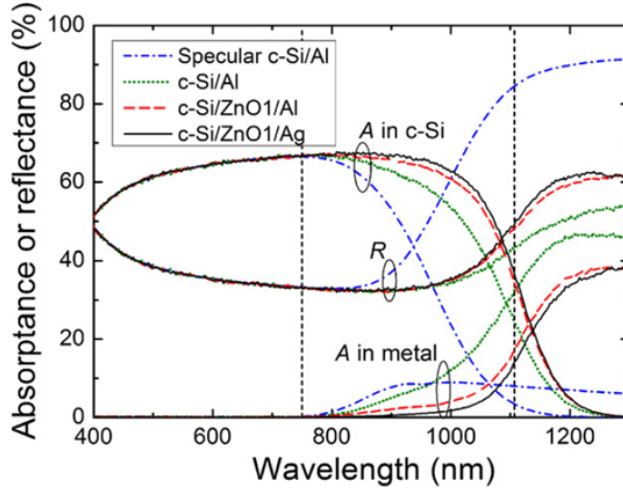
**Figure 41.** Optical constants of oxide films used in the simulation [99].

### 3.4.2.3 Results and Discussion

Figure 42 shows the calculated absorbance spectra in c-Si and metal contact and reflectance spectra for c-Si/Al, c-Si/ZnO1(200 nm)/Al, and c-Si/ZnO1(200 nm)/Ag structures with Lambertian front surface. For comparison's sake, the corresponding absorbance and reflectance spectra calculated for c-Si/Al structure with flat front surface (specular) are also included. The absorbance spectra in c-Si medium of the three structures start to deviate from each other in the wavelength longer than about 750nm, which is primarily due to the absorption in metal contact and the increased reflection especially for the structure with flat surface. From the comparison of c-Si/Al structure with flat and Lambertian surfaces, it is clear that the loss due to reflection is major cause of reduced absorption in c-Si medium. By adopting Lambertian surface, such reflection loss is greatly reduced due to increased optical path length of the incorporated light into c-Si medium. However, the absorption loss in Al contact increases in the wavelength region longer than about 970nm, which is caused by the increased absorption of light incident on Al at glancing angle. It is clear from the figure that ZnO layer between c-Si and Al contact reduces the absorption in Al contact significantly [99].

This reduction of absorption loss in Al contact is attributed to the total reflection of light impinging on the c-Si/ZnO interface at an incident angle larger than the critical angle as well as to the partial reflection of the light im-

pinging on c-Si/ZnO interface with incidence angle smaller than the critical angle. Although the overall reflectance is increased in the long wavelength region, the increment is less than the decrement of absorption loss in Al contact, leading to an increase of OAE in c-Si [99].



**Figure 42.** Simulated reflection and absorption spectra for c-Si/Al structure with specular and Lambertian surfaces and those for c-Si/ZnO1/Al and c-Si/ZnO1/Ag structure with Lambertian surfaces [99].

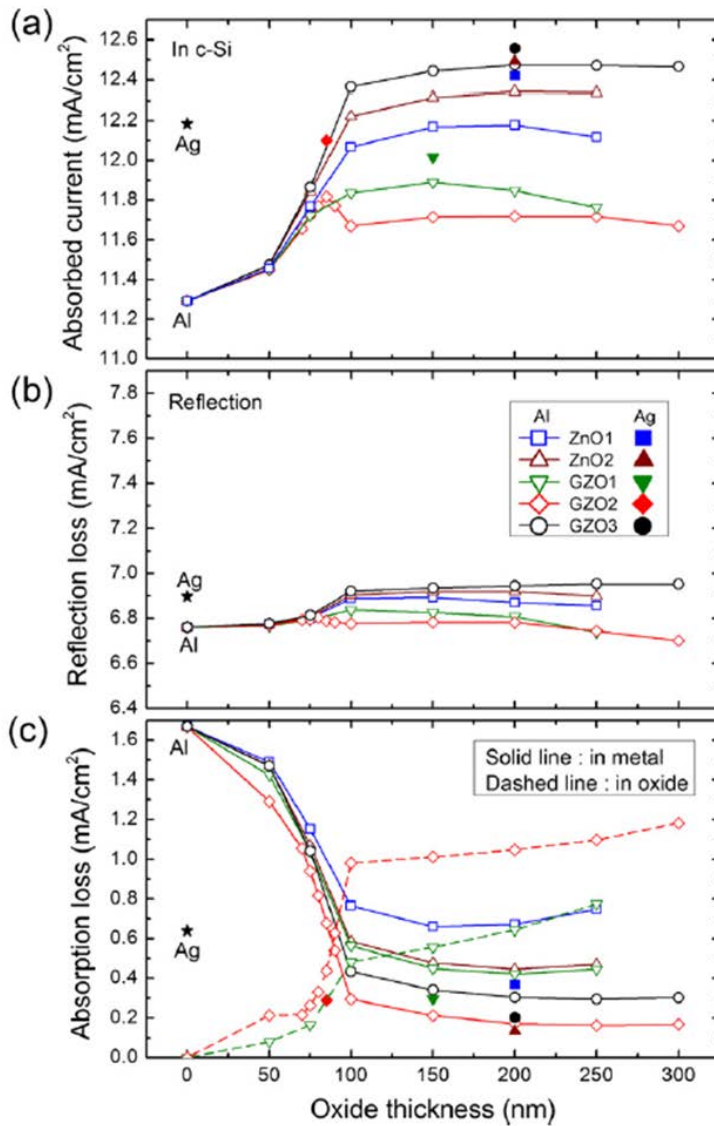
Since c-Si has band gap of 1.12 eV, the following analyses on the effect of oxide type and thickness on the absorption in c-Si medium will be carried out in the wavelength range from 750 to 1107nm, which is designated as vertical dotted lines in Figure 42. Also, the spectral intensity of solar spectrum varies with wavelength, the optically absorbed or reflected light in the wavelength range from 750 to 1107nm were converted to the amount of corresponding current using AM.1.5 Global solar spectrum [99].

In Figure 43, the current densities summed up in wavelength range of 750-1107 nm for the absorbed light in c-Si, oxide and metal back contact and the reflected light are shown as a function of oxide thickness. It is worth mentioning that the spans in y axis scale for all plots in Figure 42 are the same and that the total current density of AM1.5G solar spectrum in 750-1107 nm range is 19.72 mA/cm<sup>2</sup>. For all 5 oxide systems except for GZO2, the current due to the absorbed light in c-Si increased rapidly up to oxide thickness of 100 nm.

Above the oxide thickness of 100 nm, the variations in current densities due to the absorption in c-Si and metal contact and the reflection loss were relatively small. The highest current density was obtained from c-Si/GZO3/Al structure in which hypothetical oxide with the lowest refractive index and null extinction coefficient was used. The opposite behavior of current density is observed from the c-Si/GZO2/Al structure, in which GZO2 is the most absorptive among all the oxides examined. Unlike other systems, c-Si/GZO2/Al structure showed peak current density absorbed in c-Si at GZO2 thickness of around 85 nm [99].

This is due to very rapid increase of the absorption in highly absorptive GZO2 layer, which can be clearly seen from the Figure 43(c). Closer examination revealed that the rapidly increased absorption in GZO2 layer was caused by rapidly increasing extinction coefficient in wavelength region longer than about 900 nm. Although, for a given oxide layer with thickness above 100 nm, the thickness dependence is seen to be marginal, the maximum current density was obtained for oxide layer thickness of about 200 nm for ZnO1, ZnO2 and GZO3, while that of GZO1 being around 150 nm. The differences in current densities absorbed in c-Si for structures with ZnO1, ZnO2 and GZO3 (in decreasing order of refractive index) clearly manifest only the effect of refractive index, since these oxides are not absorbing in the interested wavelength range [99].

With decreasing refractive index, the critical angle for total internal reflection at the c-Si/oxide interface decreases, which would lead to an enhancement of OAE in c-Si. As shown in Figure 43(c), the absorption losses in Al metal contacts are in corresponding order to that of refractive index. On the other hand, the comparison between structures with GZO2 and hypothetical GZO3 oxides gives an example idea on how detrimental the absorption in oxide layer is. Due to large extinction coefficient of GZO2 film, the absorption loss increased sharply in thickness below 100 nm accompanied by a rapid decrease of the absorption loss in Al contact. Since most of the light refracted into GZO2 layer is absorbed in GZO2 film, the calculated absorption loss in Al contact is the lowest among all the examined structures [99].



**Figure 43.** Variation of the calculated current densities (a) in c-Si by absorption, (b) reflection, and (c) in metal and oxide layers as a function of oxide layer thickness. The closed symbols shown in (a) designate the current densities absorbed in c-Si for structures with Ag as metal layer [99].

For transparent conducting oxide films, the absorption in the long wavelength of visible region and near infra-red region is stemming from the free carrier absorption [93,95]. Therefore, it is inevitable that the oxide layer has to possess certain amount of carrier density in order to attain reasonable range of

resistivity. The carrier concentration of GZO1 layer was  $1.7 \times 10^{20} \text{ cm}^{-3}$ , which is not so high carrier density for transparent conducting films. In spite of moderately low carrier concentration, the absorption loss by GZO1 layer is significant [99].

From the Figure 43(b), the loss due to reflection increases slightly by addition of oxide layer in the back contact, which is also caused by the increased total reflection at the c-Si/oxide interface. But the amount of the increased reflection loss is overwhelmed by substantial decrease of the absorption loss in Al back contact. Ag is much better reflector and conductor than Al, but the use of Ag as metal back contact in commercial scale has been hindered by the long term instability [88].

In Figure 43, the current densities absorbed in c-Si and metal contact calculated using Ag contact at the oxide thickness, at which the maximum current densities were obtained by using Al contact, are shown in closed symbols. Clearly, the structures with Ag back contact gave higher OAE than those with Al back contact, and the largest difference is found between c-Si/Al and c-Si/Ag. Since Ag is much better reflector than Al, c-Si/Ag structure even without oxide layer gives much improved OAE with very low absorption loss in Ag contact. It is noted that the current density in c-Si of c-Si/Ag structure is higher than those of c-Si/ GZO1/Ag and c-Si/GZO2/Ag structures with absorptive oxide layer. Even for the structures with oxide layers without absorption loss, the enhancement in OAE when compared with the c-Si/Ag structure is not as drastic as in the case of Al back contact [99].

### 3.5 Conclusion

In this chapter, a new method for calculating the absorption energy by using the ray tracing method was introduced. This is the first algorithm to simulate the absorption energy of the thin film solar cells directly with the ray tracing method in my knowledge.

Through comparison of simple coherent case with conventional arithmetic ratio calculation, the validity of algorithm was verified. The benefit and effectiveness of new algorithm were clarified via calculation of the absorption energy in the individual layer of solar cell structure with rough surface of BSDF characteristics.

The new algorithm could yield more accurate and direct estimation of the absorption energy than the conventional method, whose calculation algorithm is limited to the use of BSDF angular profile in the form of weighting factor while assuming a flat surface.

Lastly, the new algorithm was applied for the calculation of absorption spectra of a complex system similar to a concentrator photovoltaic system, and proved to be very effective in analyzing complex geometry, which necessitates the combination of coherent and incoherent calculation.

The new solar cell simulation algorithm based on the ray tracing method is expected to provide a more realistic estimation and analysis of solar cells for the research and development of highly efficient solar cell system [29].

## **4 Simulation with FDTD Method**

### **4.1 Overview**

In the previous chapters, I proposed a fast ray tracing algorithm and a direct analysis method of absorption energy for the arbitrary textured Si solar cells of which dimension is larger than the optical wavelength or which is consisted of flat substrates only.

If the dimension of the Si solar cell system is a sub-optical wavelength or less than, the previous methods cannot be used except for the case of plane surfaces. Because, at the sub-optical wavelength system, plasma effect are not negligible. However, the ray tracing method cannot simulate this optical phenomenon. For this reason, another method, which can simulate this optical system, is needed. To simulate the system, the Finite Difference Time Domain (FDTD) method is an alternative method.

In this chapter, I propose new FDTD algorithms to simulate the absorption energy of the thin film solar cells which has a sub-optical wavelength scale. The newly developed algorithms are more efficient and fast than other conventional algorithms. These facts are validated by various numerical examples in this chapter.

### **4.2 Auto-Discretization of the System Domain**

#### **4.2.1 Algorithm**

As described in Section 1.4.1, Once the cell size has been determined, the next step is to discretize the entire system domain according to the cell size and to assign suitable material property to each cell. If the system has simple geometry and is divided into few numbers of cells, then the discretization process can be made by hands. However, in many cases, it is impossible. This is why an automatic discretization algorithm is needed.

Many automatic discretization or mesh generation algorithms have been developed in the Finite Element Method (FEM), but not in the FDTD method.

The first approach for the conventional FDTD was Y. Srisukh [46] within my knowledge. In this literature, all geometry information of the objects has to be provided in the form of mesh type by CAD system, and ray crossing and winding-number computation are used to discretize the domain automatically. However, The algorithm has an intrinsic error originated from mesh generated from curved shapes, and it does not explain how to assign material property of a boundary cell.

The second algorithm uses a linearly approximated polygon object, and this algorithm is designed for a conformal FDTD method [47]. However, this algorithm uses a conventional ray tracing method to construct a polygon approximation of an object only and uses another method to distinguish whether a cell exists inside of the object or not. This two path approach makes the discretization process complicate, and the inside check logic is somehow time consuming.

In this section, I propose an efficient and a robust algorithm for the auto-discretization process with the Slab-Outline algorithm embedded ray tracing method. The new discretization algorithm is not limited by a system geometry and a input format of geometries by which the previous algorithms are limited. With this, the discretization of the FDTD system is processed more efficiently.

All the FDTD systems consist of a boundary information, objects, sources, and free space. In the new algorithm, the boundary information is used to indicate the start and end position of a probe ray. The objects have topological and material information. The free space has material information only. With this information, I treated the FDTD system as an optical system, which can be simulated by the ray tracing method, for automatic discretization. However, in this case, the probe ray does not start at a light source or a camera, but it starts at the three surfaces (in 3-D case) of system boundary. The auto-discretization algorithm of the 3-D FDTD is as follow :

---

**Algorithm 11.** Discretize of FDTD system

---

**function** Discretize(Cell cells[], Point bounds[2], float delta[3], kd-tree rootKD)



```

//Shoot ray to +X direction at (bounds[1][X], Y, Z)
x = bounds[1][X], Ray::Direction := (1, 0, 0), prevMaterial = material of
free space
while z from bounds[1][Z] to bounds[2][Z] do
  while y from bounds[1][Y] to bounds[2][Y] do
    z = z + delta[Z] + epsilon
    y = y + delta[Y] + epsilon
    Ray::Initial := (x, y, z)
    if FindIntersection(rootKD, Ray, iP, IObject) then
      floatValue = (iP[X] - x) / delta[X]
      intValue = int(floatValue)
       $\forall$  cells [INDEX(x', y, z)]::Coef[X] := prevMaterial,  $x' \in [x, x +$ 
      intValue ]
      x = x + intValue
      if floatValue more than intValue then
        x = x + 1
        W = (floatValue - intValue) / delta[X]
        cells [INDEX(x, y, z)]::Coef[X] :=
          EffectiveMaterial( prevMaterial, IObject::MP[X], W)
      end if
      prevMaterial := IObject::MP[X]
    else
       $\forall$  cells[INDEX(x', y, z)] ::Coef[X] := material of free space,  $x' \in [x,$ 
      bounds[2][X] ]
    end if
  end while
end while

//Shoot ray to +Y direction at (X, bounds[1][Y], Z)
y = bounds[1][Y], Ray::Direction := (0, 1, 0), prevMaterial = material of
free space
** Similar to the X case

//Shoot ray to +Z direction at (X, Z, bounds[1][Z])

```

z = bounds[1][Z], Ray::Direction := (0, 0, 1), prevMaterial = material of free space

**\*\* Similar to the X case**

**end function**

---



---

### **Class 5. Cell structure of FDTD System**

---

**Class Cell**

**int BE:** { XP | XN | YP | YN | ZP | ZN }

**Coefficient Coef**[3]

**Object reference** pObject

**end Class**

---

where 'BE' is boundary flags. The XP, XN, YP, YN, ZP, and ZN indicate +X surface, -X surface, +Y surface, -Y surface, +Z surface, and -Z surface, respectively. The boundary flag is used to specify which cell is a boundary cell of the object. This flag information is used in Section 4.5.1.1. And the 'pObject' variable also is used in Section 4.5. The Coefficient is variable set of cells for the FDTD update equation. This is defined as

---

### **Class 6. Structure of Coefficient**

---

**Class Coefficient**

**int flag** { non-dispersive | dispersive | mixed }

**MaterialProperty reference** pMP

**MaterialProperty reference** pMP2

**float** Ca, Cb

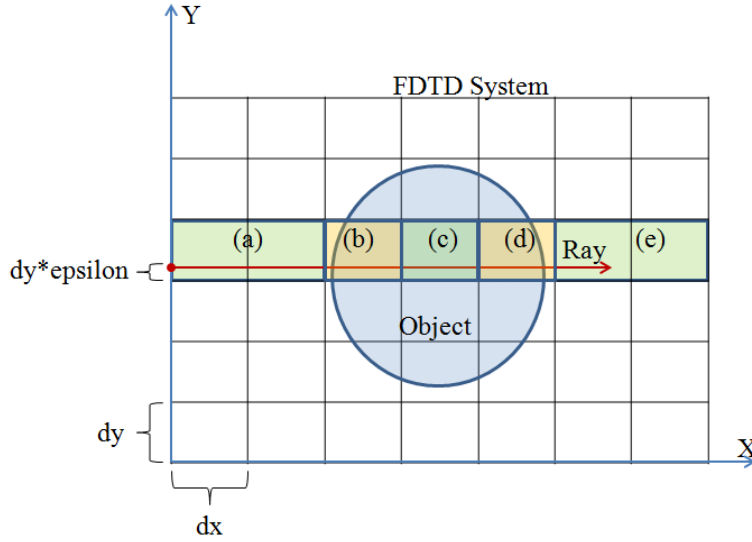
**float** Da, Db

**end Class**

---

where 'flag' is an indicator of non-dispersive, dispersive, or mixed type, MaterialProperty reference type is a pointer of the base class for material properties that are declared at Class 2. The 'Ca' and 'Cb' are used for coefficient variables of E-field update equation, for example, if Equation(1.24a) is considered,  $Ca = 1$  and  $Cb = \frac{\Delta t}{\epsilon_{i+\frac{1}{2},j,k}}$ . The 'Da' and 'Db' are used for coefficient variables of H-field update equation, for example, if Equation(1.24b) is considered,  $Da = 1$

and  $Db = \frac{\Delta t}{\mu_{i,j+\frac{1}{2},k+\frac{1}{2}}}$ . The parameter Cell array of Algorithm 11 is made by a memory allocation of a total number of the subdivisions of the FDTD system, and this is a simple operation.



**Figure 44.** Simple schema of the Algorithm 10 in 2-D case.

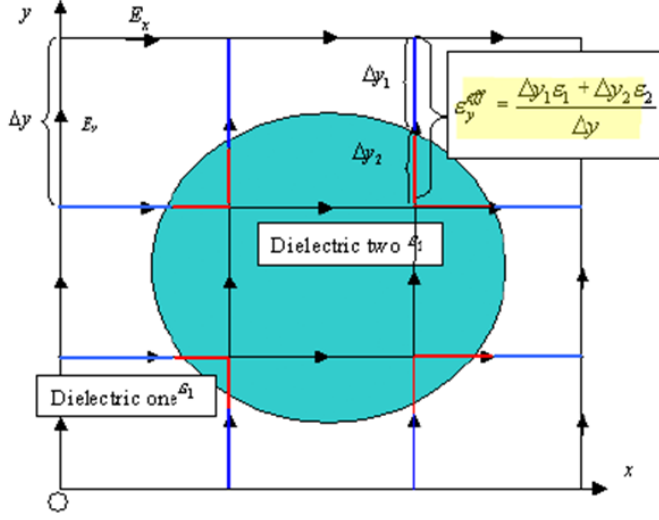
In this algorithm, the previously developed Algorithm 3 and Algorithm 5, which are used to find the valid intersection point with the object information and to get suitable material information for the divided cells in the FDTD system, are used. An initial point of the probe ray is assigned by grid point of perpendicular surface (or line, in 2-D) to the direction of the probe ray with the epsilon offset value like Figure 44. The epsilon offset value is used to prevent intersection error, which occurs as a cube type object is aligned at the system grid lines. In this thesis, I used the epsilon offset value with  $1e^{-4}$ .

The material property of the interval cells, which are located between the free space and the object or internal cells except for the object's boundary cells as Figure 44(a, c, e), are assigned at once by the material of the object from which the probe ray started. This logic increases the efficiency of Algorithm 11. The 'EffectiveMaterial' function of Algorithm 11 assigns a suitable material property to boundary cells of the intersected object. Such that, as Figure 44(b, d), if one cell is partially included in any object, it is needed to a

make decision which material property is suitable for the cell. This problem (so-called staircase problem) is an intrinsic problem of the FDTD method when it has complicated objects having curvature surfaces as a sphere.

Many solutions have been proposed to solve the staircase problem. These solutions can be categorized five groups. The first is using a non-orthogonal coordinate system instead of the original Yee's coordinate [48-51]. This method needs to change the conventional FDTD scheme entirely, so it needs extra efforts. The second is using contour path method [52-54]. This method needs to calculate volume or area of the partially filled material. The third is using a sub-grid mesh method [55-66]. Using the sub-grid mesh also modify the conventional FDTD scheme and cause a stability problem, so it needs careful treatment. The fourth is using effective permittivity [67-69]. This method does not need to modify the conventional FDTD scheme, and it generates a good accuracy result. However, it cannot treat a topological information of a boundary cell which contains different materials. The last method is conformal FDTD algorithm [47, 70-72]. It inherits the advantage of the effective permittivity method and considers the topological information suitably.

In this thesis, I followed the conformal FDTD method to circumvent the staircase problem. The conformal FDTD method for dielectric material requires the cell truncation information to calculate the effective permittivity value along the deformed cell edge, whose geometry is shown in Figure 45 [47].



**Figure 45.** Mesh truncation of a dielectric object [47].

As an example, the  $E_y$ -field update equation for the boundary cells can be written as

$$E_y^{n+1}(i, j, k) = E_y^n(i, j, k) + \frac{\Delta t}{\epsilon_y^{\text{eff}}(i, j, k) \times \Delta z} \left[ H_x^{n+\frac{1}{2}}(i, j, k) - H_x^{n+\frac{1}{2}}(i, j, k-1) \right] - \frac{\Delta t}{\epsilon_y^{\text{eff}}(i, j, k) \times \Delta x} \left[ H_z^{n+\frac{1}{2}}(i, j, k) - H_z^{n+\frac{1}{2}}(i-1, j, k) \right], \quad (4.1)$$

$$\epsilon_y^{\text{eff}}(i, j, k) = \frac{\Delta y_1 \epsilon_1 + \Delta y_2 \epsilon_2}{\Delta y}, \quad (4.2)$$

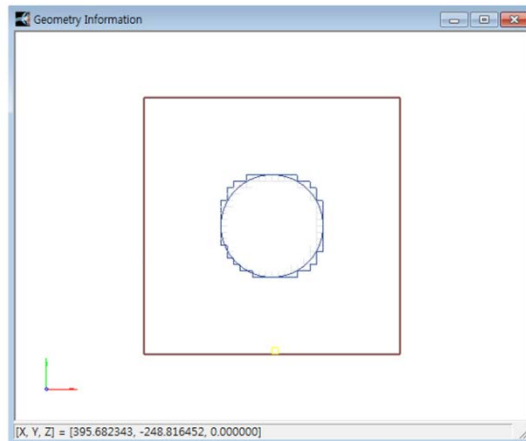
where  $\epsilon_y^{\text{eff}}(i, j, k)$  is a suitable effective permittivity of the boundary cell.

Unfortunately, as the conformal FDTD method is applied to the analysis of the thin film solar cells, the direct calculation of Equation(4.2) is impossible. Because most of the thin film solar cells are consisted of the dispersive materials, so the effective permittivity value cannot be calculated at the initial time step. To solve this problem, I developed two kinds of methods for the dispersive materials. This solution was introduced in Section 4.4.

#### 4.2.2 Results of Auto-Discretization

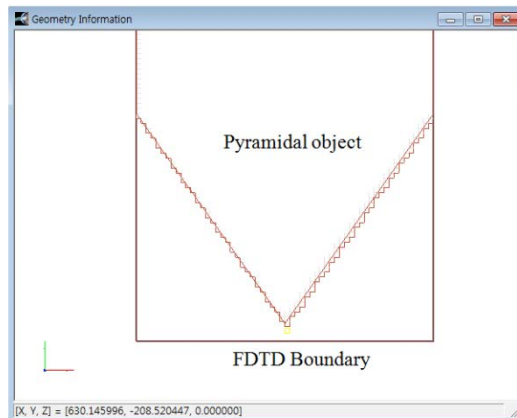
In the previous section, I proposed the efficient auto-discretization algorithm which used the Slab-Outline algorithm embedded ray tracing method. Now, I show the results in this section. It starts with two dimensional cases. At first, a

cylinder object, which has infinite length at Z-axis, was discretized. Figure 46 shows the auto-discretization result with a boundary representation.



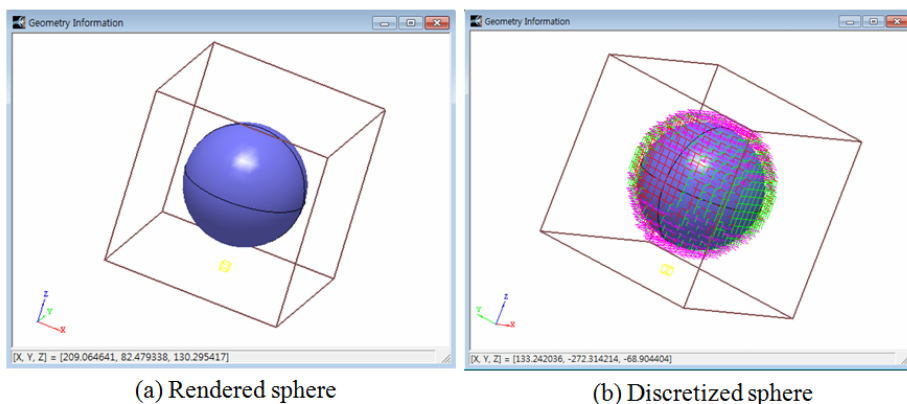
**Figure 46.** Auto-Discretization of cylinder object in 2-D.

Next, a pyramidal shape object, which also has infinite length at Z-axis, was also discretized. This result is shown in Figure 47.



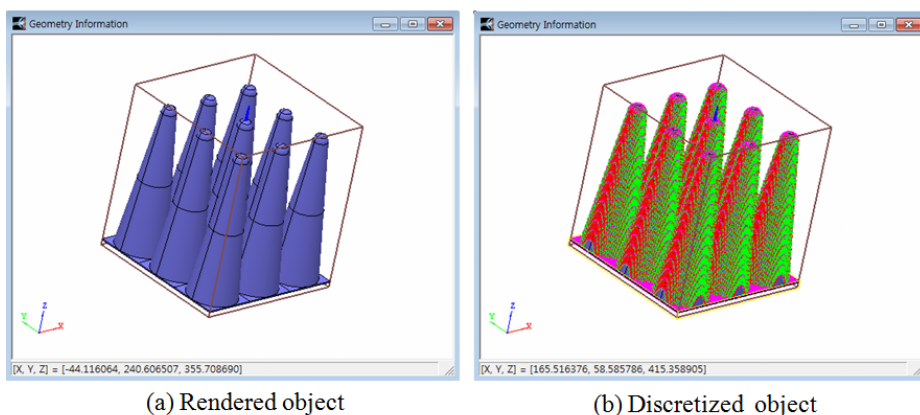
**Figure 47.** Auto-Discretization of pyramidal object in 2-D.

This auto-discretization algorithm also can make a good result in three dimensional cases. Like two dimensional case, I prepared two kinds of FDTD models and showed the discretization results with the rendered original scene for comparison. The first is a sphere shape object, and this auto-discretization result is shown in Figure 48.



**Figure 48.** Auto-Discretization of sphere object in 3-D.

The second is a moth-eye shape object array, and this result is shown in Figure 49.



**Figure 49.** Auto-Discretization of Moth-eye object in 3-D.

### 4.3 Implementation of Lorentz Model with ADE

In this section, Lorentz Model was implemented with ADE method because the thin film solar cells consist of dispersive materials, and this material has to be suitably treated at a boundary cell. Therefore, this section is a basement of the dispersive conformal FDTD algorithm introduced in the next section.

In Section 1.4.1.1-3, the brief information of the Drude, Lorentz, and Drude-Lorentz model to simulate the dispersive material with the FDTD system was introduced. As shown in Section 1.4.1.2 and 1.4.1.3, these three

models can be treated as a single Lorentz model. Therefore, I only focus on the implementation of the Lorentz model with the ADE technique. This implementation scheme is related to the 'EffectiveMaterial' function which was used at the auto-discretization algorithm in Section 4.2.

To implement Equation(1.33), (1.34), and (1.35), a static data set and dynamic data set have to be distinguished. In this, the static data set is an unchanged data set during the FDTD time-steps, and the dynamic data set keeps the updated results of the FDTD during the time-step. To save memory resource, the static data set should be saved at an object data structure because single object is divided into several cells in FDTD system. This means all cells included in the same object have the same reference object, so it can be taken these cells' static data set from the single reference object's data set. The static data structure is defined as

---

#### Class 7. Static Lorentz material parameter

---

**Class LorentzMP**

**method:**

**float** DeltaEpsilonP(**int** p)

**float** OmegaP(**int** p)

**float** DeltaP(**int** p)

**member:**

**int** NumOfPole

**float** Sigma

**float** EpsInf

**float reference** Parameters

**end Class**

---

where 'NumOfPole' is the number of Lorentz pole pairs, 'Sigma' is a conductivity, 'EpsInf' is a permittivity value when the wavelength goes to infinity, 'Parameters' is a reference type variable which is allocated when the 'NumOfPole' is assigned, and it contains three variables ( $\Delta\epsilon_p$ ,  $\omega_p$ ,  $\delta_p$ ) in a single array to prevent memory fragmentation. The functions of method section are to get the three variables ( $\Delta\epsilon_p$ ,  $\omega_p$ ,  $\delta_p$ ) easily from 'Parameters' variable. To



include the LorentzMP, the MaterialProperty of Class 2 has to be modified like this

---

### Class 8. Modified MaterialProperty structure

---

**Class** MaterialProperty

**int** flag { non-dispersive | dispersive }

**float**  $\epsilon[3]$

**float**  $\sigma[3]$

**LorentzMP reference** pLM

**end Class**

---

where 'pLM' is a reference type, such that, its memory can be allocated when it is needed.

The dynamic data set contains  $\alpha_p$ ,  $\xi_p$ ,  $\gamma_p$ , and  $\vec{J}_p(t)$ . These variables are changed during the FDTD time-steps, so these have to be declared as derived class of the Coefficient of Class 6. Therefore, the dynamic data set can be defined as

---

### Class 9. Dynamic Lorentz parameter

---

**Class** LorentzCoeff : **public** Coefficient

**method:**

**void** CalcParameter(float dt)

**void** UpdateJ(float diffE)

**float** GammaSum()

**member:**

**float** Cc

**float reference** pJ, pPreJ

**float reference** pAlpha

**float reference** pKsi

**float reference** pGamma

**end Class**

---

where 'Cc' is an additional variable for Equation(1.35c), 'pJ' is a reference variable for polarization current J, 'pPreJ' is a previous time-step value of J, and

'pAlpha', 'pKsi', 'pGamma' are reference variables for Equation(1.33a), (1.33b), and (1.33c), respectively. The variables for Equation(1.35a) and (1.35b) are inherited from the member variable 'Ca' and 'Cb' of Class 6. The functions of method section are update functions of the cell coefficients, which are related to the Lorentz parameters, according to the time-steps. Algorithm 12 is the implementation of Equation(1.33) and is defined as

---

**Algorithm 12.** Update of ADE Parameter

---

**Function** UpdateADEParam (**float** dt)

```

int p := 1
while p <= Number of Poles do
    deltaP := pMP::pLM::DeltaP(p)
    omegaP := pMP::pLM::OmegaP(p)
    dEpsP := pMP::pLM::DeltaEpsilonP(p)
    One_DeltaDt := 1 + deltaP * dt
    pAlpha[p] := (2 - omegaP^2 * dt^2) / One_DeltaDt
    pKsi[p] := (deltaP * dt - 1) / One_DeltaDt
    pGamma[p] := ( $\epsilon_0$  * dEpsP * omegaP^2 * dt^2) / One_DeltaDt
    p := p + 1
end while

```

**end Function**

---

where 'dt' is the time interval for field update. Algorithm 13 is the implementation of Equation (1.34). This is defined like this

---

**Algorithm 13.** Update of Polarization Current

---

**Function** UpdateJp(**float** diffE)

```

int p := 1
while p <= Number of Poles do
    preJ := pJ[p]
    pJ[p] := pAlpha[p] * pJ[p] + pKsi[p] * pPreJ[p] + pGamma[p] * diffE
    pPreJ[p] := preJ
    p := p + 1
end while

```

**end while**

**end Function**

---

where 'diffe' is the third part (*i.e.*,  $\frac{\vec{E}_p^{n+1} - \vec{E}_p^{n-1}}{2\Delta t}$ ) of the right hand side of Equation(1.34). Algorithm 14 is the implementation of  $\frac{1}{2} \sum_{p=1}^N \gamma_p$  part of Equation(1.35). This is defined like this

---

**Algorithm 14.** Summation of Gamma value

---

**Function** GammaSum( )

**int** p := 1

**float** sum := 0.0

**while** p <= Number of Poles **do**

    sum := sum + pMP::pLM::pGamma[p]

    p := p + 1

**end while**

sum := 0.5 \* sum

**return** sum

**end Function**

---

#### 4.4 EffectiveMaterial Function

In Chapter 1, I introduced five kinds of algorithms to diminish the staircase errors in curved boundary problems. As mentioned in Chapter 1, these algorithms have advantages and disadvantages each other. Among them, the conformal FDTD is more stable than others and require less modification of the conventional update equations of the FDTD than others. Therefore, the newly developed algorithm also follows the concept of the conformal FDTD method.

In this section, two kinds of algorithms are introduced for curved dispersive material boundary condition and compared these with the staircase method. The first is Round-Off algorithm which is similar to the staircase method, except for it has an anisotropic material property. The second is a dispersive conformal FDTD(D-CFDTD) algorithm for the curved dispersive

material boundary condition with Lorentz model. The conventional conformal FDTD algorithm for curved pure dielectric and perfect conductive material was introduced by several literatures [70 - 72]. However, they did not provide a solution for the dispersive material.

The new algorithms follow the concept of the conformal FDTD method. Therefore, they keep the advantages of the conformal FDTD method, moreover the Round-Off algorithm is more simple and efficient for the Drude term dominant materials than the original conformal FDTD method and the Dispersive conformal FDTD (D-CFDTD) algorithm enable to apply the conformal FDTD method to dielectric dispersive materials with the conventional ADE method. The effectiveness and correctness of these algorithms is validated by comparing the simulated scattering ratios with different spatial delta values with an exact Mie solution.

#### 4.4.1 Round-Off Algorithm

The Round-Off algorithm starts with a same concept of the conformal FDTD algorithm. However, this does not use a weight variable which has continuous real value from 0 to 1, but this chooses zero or one value according to a barrier of the filling ratio of the first material, in this thesis, 0.5 was used as the barrier value. Such that,

$$\begin{cases} \frac{\Delta L_1}{\Delta L} < \beta, & \epsilon_L^{\text{eff}}(i, j, k) = \epsilon_2 \\ \frac{\Delta L_1}{\Delta L} \geq \beta, & \epsilon_L^{\text{eff}}(i, j, k) = \epsilon_1 \end{cases} : L \in \{x, y, z\}, \quad (4.3)$$

where  $\beta$  is a barrier value,  $\epsilon_L^{\text{eff}}(i, j, k)$  is an effective permittivity of the cell at  $(i, j, k)$  grid index,  $\Delta L$  is cell dimensions, and  $\Delta L_1$  is an embedded length of the first material at each axis direction. Therefore, the Round-Off algorithm applied the EffectiveMaterial function is as this :

---

#### Algorithm 15. Round-Off method for EffectiveMaterial

---

**function** EffectiveMaterial(**MaterialProperty** preM, **MaterialProperty** curM, **float** w1)

**Coefficient reference** pCoef

$\beta = 0.5$

```

if w1 <  $\beta$  then
    if curM::flag equal non-dispersive then
        pCoef := new Coefficient
        pCoef::flag = curM::flag
        pCoef::pMP := curM
    else
        pCoef := new LorentzCoef
        pCoef::flag = curM::flag
        pCoef::pMP := curM
    end if
else
    if preM::flag equal non-dispersive then
        pCoef := new Coefficient
        pCoef::flag = preM::flag
        pCoef::pMP := preM
    else
        pCoef := new LorentzCoef
        pCoef::flag = preM::flag
        pCoef::pMP := preM
    end if
end if
return pCoef
end function

```

---

#### 4.4.2 Dispersive Conformal FDTD (D-CFDTD) Algorithm

Equation(4.1) and (4.2) are E-field update equation for the conventional conformal FDTD algorithm which can diminish the staircase problem of curved pure dielectric object. However, these equations are not applied to the curved dispersive object directly because the permittivity of dispersive object changes according to the frequency of the input source. This property was explained in Chapter 1.

To apply the conformal FDTD algorithm to the dispersive boundary, Equation(4.1) need to be changed to the form of Equation(1.35). The applied equation is shown as

$$\vec{E}^{n+1} = \bar{\bar{C}}_1 \vec{E}^{n-1} + \bar{\bar{C}}_2 \vec{E}^n + \bar{\bar{C}}_3 \left\{ \nabla \times \vec{H}^{n+1/2} - \frac{1}{2} \left[ \bar{\bar{W}}_1 \sum_{p=1}^{N_1} \left[ (1 + \alpha_1) \vec{J}_{1p}^n + \xi_1 \vec{J}_{1pn-1} + W_{2p=1N2}(1+\alpha_2) \vec{J}_{2pn} + \xi_2 \vec{J}_{2pn-1} \right] \right] \right\} \quad (4.4)$$

where

$$\bar{\bar{C}}_1 = \begin{bmatrix} \frac{\gamma'_{pxx}}{2\varepsilon_0\varepsilon_{\infty xx} + \gamma'_{pxx} + \sigma'_{xx}\Delta t} & 0 & 0 \\ 0 & \frac{\gamma'_{pyy}}{2\varepsilon_0\varepsilon_{\infty yy} + \gamma'_{pyy} + \sigma'_{yy}\Delta t} & 0 \\ 0 & 0 & \frac{\gamma'_{pzz}}{2\varepsilon_0\varepsilon_{\infty zz} + \gamma'_{pzz} + \sigma'_{zz}\Delta t} \end{bmatrix}, \quad (4.4a)$$

$$\bar{\bar{C}}_2 = \begin{bmatrix} \frac{2\varepsilon_0\varepsilon_{\infty xx} - \sigma'_{xx}\Delta t}{2\varepsilon_0\varepsilon_{\infty xx} + \gamma'_{pxx} + \sigma'_{xx}\Delta t} & 0 & 0 \\ 0 & \frac{2\varepsilon_0\varepsilon_{\infty yy} - \sigma'_{yy}\Delta t}{2\varepsilon_0\varepsilon_{\infty yy} + \gamma'_{pyy} + \sigma'_{yy}\Delta t} & 0 \\ 0 & 0 & \frac{2\varepsilon_0\varepsilon_{\infty zz} - \sigma'_{zz}\Delta t}{2\varepsilon_0\varepsilon_{\infty zz} + \gamma'_{pzz} + \sigma'_{zz}\Delta t} \end{bmatrix}, \quad (4.4b)$$

$$\bar{\bar{C}}_3 = \begin{bmatrix} \frac{2\Delta t}{2\varepsilon_0\varepsilon_{\infty xx} + \gamma'_{pxx} + \sigma'_{xx}\Delta t} & 0 & 0 \\ 0 & \frac{2\Delta t}{2\varepsilon_0\varepsilon_{\infty yy} + \gamma'_{pyy} + \sigma'_{yy}\Delta t} & 0 \\ 0 & 0 & \frac{2\Delta t}{2\varepsilon_0\varepsilon_{\infty zz} + \gamma'_{pzz} + \sigma'_{zz}\Delta t} \end{bmatrix}, \quad (4.4c)$$

$$\gamma'_{pLL} = \frac{1}{2} \left( W_{1LL} \cdot \sum_{p=1}^{N_1} \gamma_{1p} + W_{2LL} \cdot \sum_{p=1}^{N_2} \gamma_{2p} \right), L \in \{x, y, z\}, \quad (4.4d)$$

$$\varepsilon_{\infty LL} = W_{1LL} \cdot \varepsilon_{1\infty} + W_{2LL} \cdot \varepsilon_{2\infty}, L \in \{x, y, z\}, \quad (4.4e)$$

$$\sigma'_{LL} = W_{1LL} \cdot \sigma_1 + W_{2LL} \cdot \sigma_2, L \in \{x, y, z\}, \quad (4.4f)$$

$$W_{1LL} = \frac{\Delta L_1}{\Delta L}, L \in \{x, y, z\}, \quad (4.4g)$$

$$W_{2LL} = 1 - W_{1LL}, L \in \{x, y, z\}. \quad (4.4h)$$

In Equation(4.4), the  $\varepsilon_{1\infty}$ ,  $\varepsilon_{2\infty}$ ,  $\sigma_1$ , and  $\sigma_2$  are constant value and time independent, so  $\varepsilon_{\infty LL}$  and  $\sigma'_{LL}$  are also constant value and time independent. To implement Equation(4.4), a new data structure derived from Class 9 was made as follow :

---

**Class 10. Mixed Lorentz boundary**

---

**Class** LorentzCoeff BC : **public** LorentzCoef

**method:**

**void** CalcParameter(float dt)

**void** UpdateJ(float diffE)

**float** GammaSum()

**member:**

**float** W

**float reference** pJ2, pPreJ2

**float reference** pAlpha2

**float reference** pKsi2

**float reference** pGamma2

**end Structure**

---

where the extended reference variables are dynamic data set for the second material included in the boundary cell. Also, I modified Algorithm 12, Algorithm 13, and Algorithm 14 like these

---

**Algorithm 16. Update of ADE Parameter for D-CFDTD**

---

**Function** UpdateADEParam (**float** dt)

**int** p := 1

**while** p <= pMP::pLM::NumOfPole **do**

deltaP := pMP::pLM::DeltaP(p)

omegaP := pMP::pLM::OmegaP(p)

dEpsP := pMP::pLM::DeltaEpsilonP(p)

One\_DeltaDt := 1 + deltaP \* dt

pAlpha[p] := (2 - omegaP^2 \* dt^2) / One\_DeltaDt

pKsi[p] := (deltaP \* dt - 1) / One\_DeltaDt

pGamma[p] := ( $\epsilon_0$  \* dEpsP \* omegaP^2 \* dt^2) / One\_DeltaDt

p := p + 1

**end while**

p := 1

```

while p <= pMP2::pLM::NumOfPole do
    deltaP := pMP2::pLM::DeltaP(p)
    omegaP := pMP2::pLM::OmegaP(p)
    dEpsP := pMP2::pLM::DeltaEpsilonP(p)
    One_DeltaDt := 1 + deltaP * dt
    pAlpha2[p] := (2 - omegaP^2 * dt^2) / One_DeltaDt
    pKsi2[p] := (deltaP * dt - 1) / One_DeltaDt
    pGamma2[p] := ( $\epsilon_0$  * dEpsP * omegaP^2 * dt^2) / One_DeltaDt
    p := p + 1
end while
end Function

```

---



---

**Algorithm 17.** Update of Polarization Current for D-CFDTD

---

```

Function UpdateJp(float diffE)
    int p := 1
    while p <= pMP2::pLM::NumOfPole do
        preJ := pJ[p]
        pJ[p] := pAlpha[p] * pJ[p] + pKsi[p]*pPreJ[p] + pGamma[p]*diffE
        pPreJ[p] := preJ
        p := p + 1
    end while
    p := 1
    while p <= pMP2::pLM::NumOfPole do
        preJ := pJ2[p]
        pJ2[p] := pAlpha2[p] * pJ2[p] + pKsi2[p]*pPreJ2[p] +
            pGamma2[p]*diffE
        pPreJ2[p] := preJ
        p := p + 1
    end while
end Function

```

---



---

**Algorithm 18.** Summation of Gamma Values

---

**Function** GammaSum( )**int** p := 1**float** sum := 0.0**float** W2 := 1.0 - W**while** p <= pMP::pLM::NumOfPole **do**

sum := sum + W \* pMP::pLM::pGamma[p]

p := p + 1

**end while**

p := 1

**while** p <= pMP2::pLM::NumOfPole **do**

sum := sum + W2 \* pMP::pLM2::pGamma[p]

p := p + 1

**end while**

sum := 0.5 \* sum

**return** sum**end Function**

---

Until now, I proposed the dispersive conformal FDTD (D-CFDTD) algorithm for the dispersive curved boundary. When the D-CFDTD algorithm is applied to the 'EffectiveMaterial' function, three different boundary cases have to be considered. The first case is that the two boundary materials are both non-dispersive materials. The second case is that the two boundary materials are both dispersive materials. The third case is that the two boundary materials are different types of materials such that the one is dispersive, and the other is non-dispersive materials.

The first case is the simplest case. In this case, 'EffectiveMaterial' has the same logic of Equation(4.2) which is the conventional conformal FDTD algorithm [47, 70-72]. The second and third cases need the dispersive conformal FDTD(D-CFDTD) algorithm. However, if the dispersive material is Drude term dominant or has Drude term only, then another method is needed which

is similar to a staircase method or an S-EP method because, in this case, surface plasmon effect is more crucial [75].

The dispersive conformal FDTD(D-CFDTD) algorithm applied the 'EffectiveMaterial' function is as follow :

---

**Algorithm 19.** Dispersive Conformal FDTD method for EffectiveMaterial

---

```

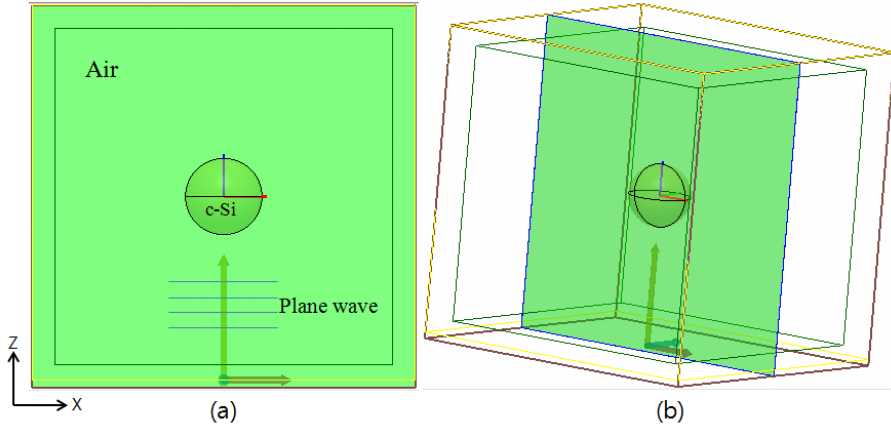
function EffectiveMaterial(MaterialProperty preM, MaterialProperty
curM, float w1)
    Coefficient reference pCoef
    if preM::flag equal non-dispersive and preM::flag equal curM::flag then
        pCoef := new Coefficient
        pCoef::flag = non-dispersive
        Apply Equation(4.2)
    else if preM and curM have no Drude term then
        pCoef := new LorentzCoefBC
        pCoef::flag = mixed
        pCoef::pMP := preM
        pCoef::pMP2 := curM
        pCoef::W := w1
    else
        Apply staircase or S-EP algorithm
    end if
    return pCoef
end function

```

---

#### 4.4.3 Validation

In the previous section, I proposed the Round-Off and the dispersive conformal FDTD(D-CFDTD) algorithms to solve the dispersive curved boundary problem.



**Figure 50.** Schematic illustration of the crystalline silicon sphere in free space. (a) is 2D projection view. (b) is 3D view. The green plane is detector to monitor field values. The outer box is system boundary and the inner box is far-field detector.

Now, I validate these algorithms using Mie theory for the scattering of a crystalline silicon(c-Si) sphere. For this purpose, a simple system, which is consisted of a crystalline silicon(c-Si) sphere with 100nm diameter and plane wave source with 600nm wavelength, and the sphere is located in free space(*i.e.*, Relative permittivity value is 1.0), was prepared. The FDTD configuration of this system was depicted in Figure 50.

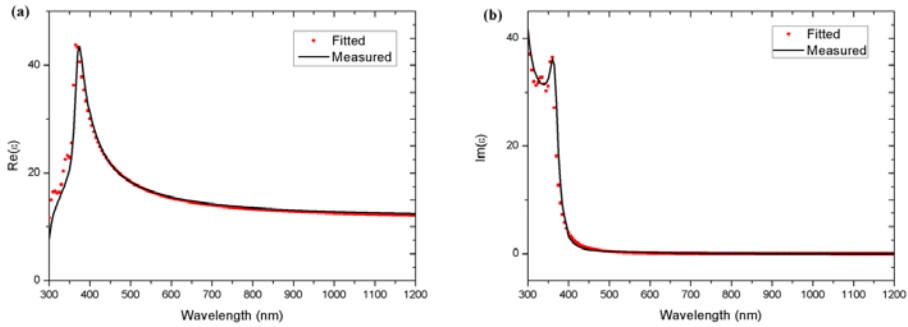
Next, the Lorentz parameter values listed in Table 3 was prepared. To find these parameters, the Levenberg-Marguardt (LM) algorithm [100] and five Lorentz pole pairs were used. In fact, the five Lorentz pole pairs are not the optimal number of pole pairs in the optical and the infrared wavelength ranges. In these ranges, A. Deinega et al. fitted c-Si permittivity values with only two pole pairs using their modified Lorentz model [101]. However, this model is not proved to fit other materials, so I decided to find my Lorentz parameters of c-Si for 300 - 1200nm wavelength ranges.

Pole No.	Delta Epsilon	Plasma Frequency	Damping Frequency
1	3.095913E+00	5.541390E+15	3.553230E+14
2	-6.272547E+00	6.088290E+15	1.157100E+15
3	1.223609E+00	5.175160E+15	1.287820E+14
4	1.000000E+01	6.329490E+15	6.133380E+14
5	2.368657E+00	7.171270E+15	3.773990E+13

Table 3. Lorentz parameters of c-Si.

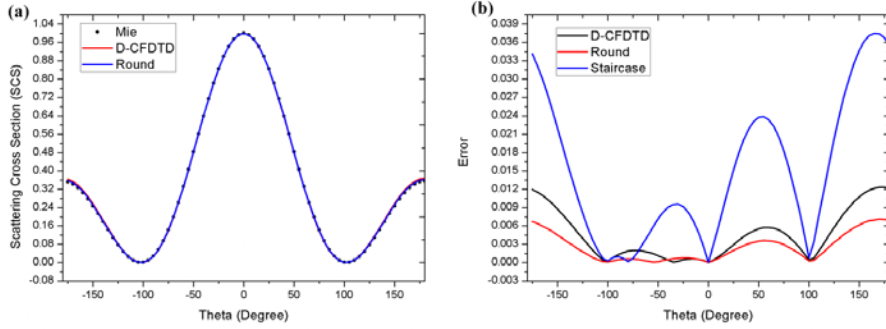
where epsilon infinity( $\epsilon_\infty$ ) is 1.0.

In Figure 51, the newly fitted values are plotted with the measured complex permittivity values of c-Si. In the 350 - 1200nm wavelength ranges, the fitted values are exactly matched with the measured one. However, below the 350nm wavelength, the fitted values slightly mismatch with the measured one but this difference can be negligible.



**Figure 51.** (a) Real and (b) Imaginary permittivity fitting values of the c-Si are compared with experimental data.

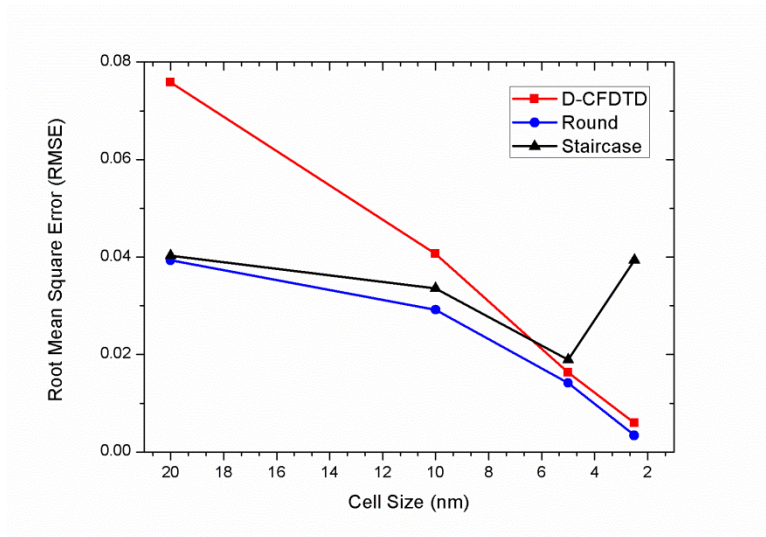
With this targeting model, the correctness and the convergence of the newly developed algorithms is validated. For the correctness, the FDTD system was discretized to an evenly divided sub-grids with 2.5nm cell size in all x, y, z directions and was surrounded by CPML absorption boundary condition [78] to prevent suspicious reflection waves. To keep stability condition, the Courant number is set to 0.9, and maximum time step is set to 40,000. One period sinusoidal input source impinged with the full scattering field mode [104]. The simulated data was plotted with Mie data at Figure 52(a) and their absolute error values also was plotted at Figure 52(b).



**Figure 52.** Comparison of Round-Off and D-CFDTD with Mie theory. (a) is an angular scattering cross section(SCS) values. (b) is absolute errors.

As shown in Figure 52(a), the FDTD results were exactly matched with the exact Mie data except for the around start and end angles of around  $-180^\circ$ . At this angle, the detected values were all back scattering values, but the simulation model is forward scattering dominant system so this may cause these errors. In Figure 52(b), absolute errors of the three kinds of algorithms were compared. The first is the dispersive conformal FDTD (D-CFDTD), the second is the Round-Off (Round), and the last is the staircase algorithm. As Figure 52(b) shown, the Round-Off algorithm had the smallest error among them, and next is the D-CFDTD algorithm. Also, these two algorithms' errors were about three times less than the staircase's error in all scattering angles. From the result, the correctness and the superior of the round-off and D-CFDTD algorithms was validated.

Next, a convergence of the algorithms was checked through various cell sizes. For this purpose, four kinds of cell sizes (*e.g.*, 20, 10, 5, and 2.5 nm) with three kinds of algorithms (*e.g.*, Round-Off, D-CFDTD, and Staircase) were simulated, and root mean square errors (RMSE) of these cases were calculated. Their results were plotted in Figure 53.



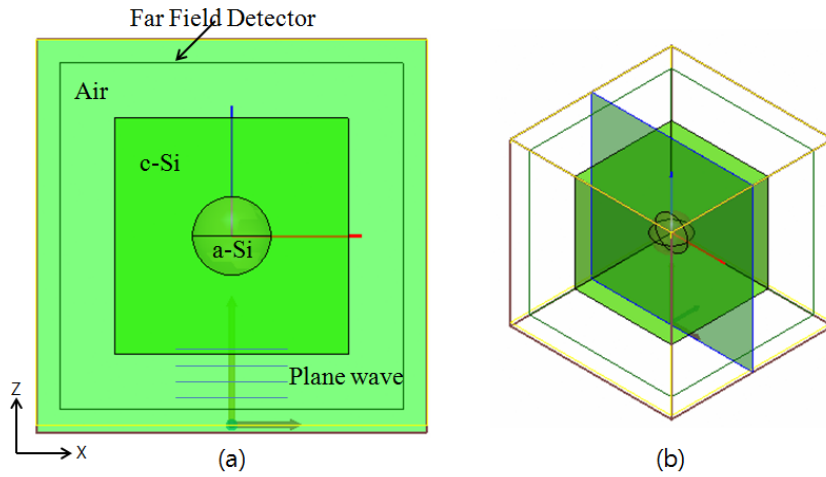
**Figure 53.** Root mean square errors of the three kinds of algorithms.

From these results, the convergence of the Round-Off and D-CFDTD algorithms was confirmed. Also, I found that the staircase algorithm cannot simulate a curved dispersive material precisely because the errors oscillated around the 5nm cell size. Therefore, in this case, it is needed the Round-Off and the D-CFDTD algorithms to find an exact solution.

In this section, the algorithms were validated with a simple model. At the next section, more complex models were researched with the Round-Off and the D-CFDTD algorithms.

#### 4.4.4 Numerical Analysis

In this section, I simulated a boundary cell model with two kinds of dispersive materials. The first material is an amorphous silicon (a-Si) sphere with 100nm diameter and the second material is a crystalline silicon (c-Si) cube whose all edges are 300nm dimension. The a-Si sphere embedded in the c-Si cube, and other conditions are same as the previous model. This model was depicted in Figure 54.



**Figure 54.** Schematic illustration of the a-Si sphere immersed c-Si cube in free space. (a) is 2D projection view. (b) is 3D view.

This configuration is frequently used to enhance the absorption efficiency in the thin film solar cells [103]. In this case, the accurate detection of the boundary of the two different materials and the decrease of the staircase error are necessary to obtain an accurate simulation result because the two materials are immersed with each other and they are all dispersive materials with curved boundary. Therefore, this model was one of the suitable examples in which I could confirm the effectiveness of my auto-discretization algorithm and the two previous algorithms.

**(a)**

Pole No.	Delta Epsilon	Plasma Frequency	Damping Frequency
1	6.180692E+00	5.723150E+15	9.092160E+14
2	-1.200000E+00	2.615290E+15	5.889290E+14
3	2.947256E+00	4.774260E+15	7.087220E+14
4	4.868531E+00	7.175570E+15	1.609500E+15

**(b)**

Pole No.	Delta Epsilon	Plasma Frequency	Damping Frequency
1	-1.200000E+00	2.602460E+15	6.685630E+14
2	4.781800E+00	6.259470E+15	9.909940E+14
3	4.400120E+00	5.342100E+15	7.325310E+14
4	1.926820E+00	4.503200E+15	6.146310E+14
5	2.861590E+00	8.104770E+15	1.740870E+15

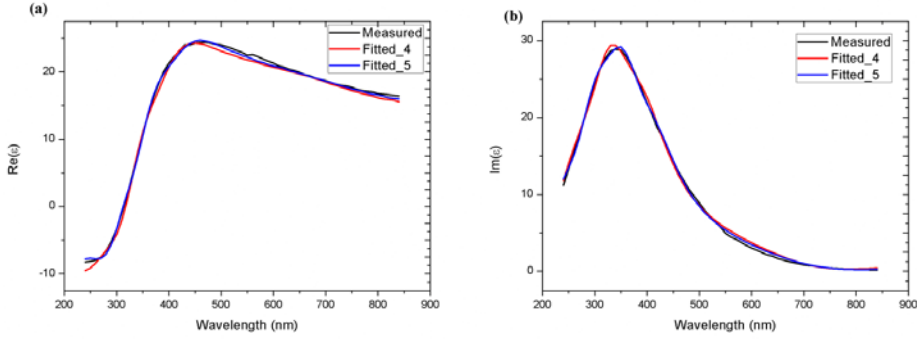
**Table 4. Lorentz parameters of a-Si, (a) is four pole pairs model and (b) is**

### five pole pairs model.

where epsilon infinity( $\epsilon_{\infty}$ ) is 1.0.

Lorentz parameters of the amorphous silicon (a-Si) sphere is also needed to simulate this model. To get the values, I fitted two kinds of Lorentz parameter sets with four pole pairs and five pole pairs, respectively, and each parameter sets were listed in Table 4(a) and (b). The experiment data of a-Si in frequency domain came from the SOPRA nk database [102].

As shown in Figure 55, the five pole pairs model is more accurate than the four pole pairs. However, their difference of accuracy was negligible in this model, so I used the four pole pairs model in this simulation to reduce simulation time.



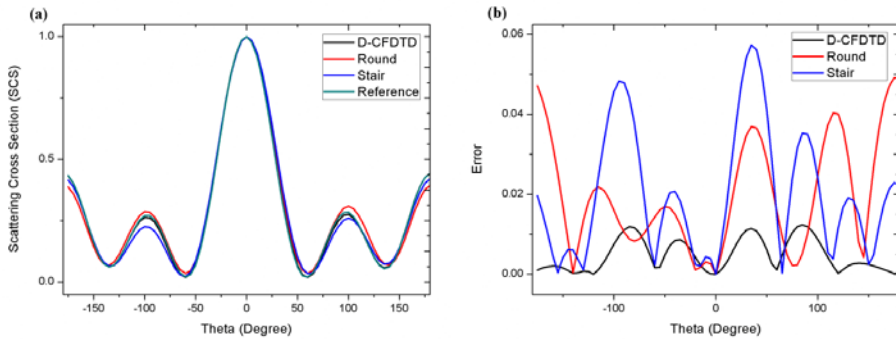
**Figure 55.** (a) Real and (b) Imaginary permittivity fitting values of the a-Si are compared with experimental data.

For this model, an exact Mie scattering data could not be calculated, so another reference data was needed for error analysis. For this purpose, I discretized the system domain with a same 2.5nm cell size at all axis and simulated with the D-CFDTD algorithm because the D-CFDTD algorithm converged to the exact solution with negligible error at the 2.5nm cell size was validated in the previous section. To compare with this reference data, I discretized the system with a same 5nm cell size at all axis and simulated with the three kinds of algorithms as the previous section.

The simulation results of the three different algorithms (*e.g.*, D-CFDTD, Round-Off, staircase) were plotted in Figure 56. If Figure 52(b) and Figure



56(b) are compared with each other, then it could be easily found that the error tendency of the Round-Off algorithm changes dramatically between the two models. Even if it was considered that the reference data was made by the D-CFDTD algorithm. However, the D-CFDTD algorithm kept its tendency between the two models. Of course, the error of the Round-Off algorithm is less than the one of the staircase algorithm in both models.



**Figure 56.** Comparison of Round-Off and D-CFDTD with more fine mesh D-CFDTD. (a) is an angular scattering cross section(SCS) values. (b) is absolute errors.

According to the results of the previous section and this, it can be concluded that the Round-Off and the D-CFDTD algorithms are accurate and efficient. Especially, the D-CFDTD algorithm is suitable for most of the curved dispersive materials. However, when a Drude term dominant dispersive material is treated, this algorithm cannot be used because the D-CFDTD algorithm is derived from the CFDTD which is suitable for dielectric materials.

## 4.5 Simulation of Absorption Energy

The precise calculation of the absorption energy of the thin film solar cells is very important to measure the enhancement of efficiency of the thin film solar cells as explained in Chapter 3.

To calculate absorbed energy in the nanometer scale thin film solar cells, many researchers have approached with mainly three kinds of methods. The first is an indirect method such that the absorption energy was calculated from the reflected and the transmitted energies with this equation

$$A = 1 - T - R, \quad (4.5)$$

where A is absorbed energy, T is transmitted energy, and R is reflected energy [105 - 110]. With this method, only the absorption energy of the entire system can be assumed without each layer's absorption energy, and this method has one assumption that all the absorbed optical energy in the entire system will be converted to electric energy in the active layer. However, this is not true as explained in Chapter 3.

The second method is a direct method which calculate the absorption energy from Poynting theorem like this :

$$\text{Absorption Energy} = \iiint (\frac{1}{2} |\epsilon''| E^2(\omega)) dV, \quad (4.6)$$

where  $\epsilon''$  is the imaginary part of complex permittivity values and  $\omega$  is the angular frequency. This equation has an assumption of  $\mu'' = 0$  [39, 111-113]. This method allows to calculate the absorption energy of the individual layers of the thin film solar cells. However, this method needs much extra volume memory to keep all volume cells' electric field values that are used at the discrete Fourier transform process and at the last volume integration, and this extra memory needs extra computational time. Also, it sometimes generates wrong result at high frequency ranges as shown in Figure 62.

The third method is also a direct method. It starts from the divergence part of the Poynting theorem [114,115]. However, James R. Nagel et al. applied the divergence theorem only to a flat layer systems with a power transmission coefficient at depth z, such that their absorption energy equation is

$$A(z, \lambda) = T(0, \lambda) - T(z, \lambda), \quad (4.7a)$$

$$A_{\text{flux}}(z, \lambda) = A(z, \lambda) \Phi_0(\lambda), \quad (4.7b)$$

where  $A(z, \lambda)$  is absorption factor,  $T(z, \lambda)$  is a power transmission coefficient through an xy-plane at depth z,  $A_{\text{flux}}(z, \lambda)$  is the flux absorbance, and  $\Phi_0(\lambda)$  is the AM-1.5 spectrum [114]. Therefore, with this method, the absorption energy of an arbitrary shaped object cannot be calculated. Wei Wang et al. suggested another general divergence equation to calculate absorption energy as

$$\text{Absorption} = \oint \vec{S}(\vec{r}, \omega) \cdot d\vec{a} \quad (4.8)$$

where  $\vec{S}(\vec{r}, \omega)$  is the Poynting vector [116]. Nevertheless, they also applied this equation to the only flat Si layer. This limitation came from the difficulty of the extraction for the continuous boundary of arbitrary shaped object.

In the following section, I proposed a novel direct absorption energy calculation algorithm with a continuous boundary extraction algorithm using the divergence part of the Poynting theorem. With this algorithm, the absorption energy of the arbitrary shaped object embedded thin film solar cell can be simulated with the smallest memory efficiently. These facts were validated by two kinds of numerical examples.

#### 4.5.1 Algorithm

I started with the Poynting theorem which explains energy conservation for the electromagnetic field [116].

$$\iiint (\mathbf{H} \cdot \frac{\partial \mathbf{B}}{\partial t} + \mathbf{E} \cdot \frac{\partial \mathbf{D}}{\partial t} + \mathbf{E} \cdot \mathbf{J}) dV = - \oint (\mathbf{E} \times \mathbf{H}) \cdot d\mathbf{S}, \quad (4.9)$$

where bold character means complex vector notation.

The first and the second terms of the left hand side of Equation(4.9) are energy storage per unit volume for a magnetic and an electric field, respectively, and the third term means ohmic power loss [116]. The right hand of Equation(4.9) is an input(or output) energy which penetrated through the closed surface area. If it is assumed that a single electromagnetic pulse impinged at a material and flowed enough time, then some of the stored electric and magnetic field energies flow out the material through its surface area. This flow out energy is the real permittivity and permeability part of the stored electric and magnetic field energies. In the last, some of stored energy of the other part and ohmic power loss are absorbed into the material.

Therefore, if the output Poynting power is subtracted from the input Poynting power throughout the unit surface normal, the total absorption energy can be calculated from the surface integration of the difference of the Poynting powers. This is a main concept to calculate the absorption energy with the divergence part of the Poynting theorem.

#### 4.5.1.1 Extract Object's Continuous Boundary

This algorithm is deeply related to the auto-discretization algorithm which was introduced in Section 4.2. To identify boundary of an object, an intersection information is needed, which contains where is the intersection point and which objects are sharing its boundary. Moreover, the FDTD system has discretized cubic cells instead of the real objects, so it is needed to adjust the real object's boundary to the suitable cells. Sometimes, real object's boundary can be located into an internal position of one cell. In this case, the 'EffectiveMaterial' function is needed to reduce staircase error.

The 'Cell' class(defined in Class 5) has the 'BE' member variable that is used to identify which surfaces of the cell are used to represent the object's boundary and the 'pObject' variable that is used to keep a reference value of the object which include the cell and this 'Cell' information is set at the 'Discretize' function (*i.e.*, Algorithm 11) and is adjusted at an 'MakeContinuousBoundary' function which is defined at next. Algorithm 11 have to be slightly changed to write the boundary information as follow :

---

**Algorithm 20.** Modified Discretize method for FDTD System

---

**function** Discretize(Cell cells[], Point bounds[2], float delta[3], KD-tree rootKD)

    //Shoot ray to +X direction at (bounds[1][X], Y, Z)

    x = bounds[1][X]; Ray::Direction := (1, 0, 0);

    prevMaterial := material of free space

    prevObject := NULL

**while** z from bounds[1][Z] to bounds[2][Z] **do**

**while** y from bounds[1][Y] to bounds[2][Y] **do**

            z = z + delta[Z] + epsilon

            y = y + delta[Y] + epsilon

            Ray::Initial := (x, y, z)

            prevObject := Object in which ray start.

**if** FindIntersection(rootKD, Ray, iP, IObject) **then**

                floatValue = (iP[X] - x) / delta[X]

```

intValue = int(floatValue)
 $\forall$  cells [INDEX(x', y, z)]::Coef[X]
    := prevMaterial, x'  $\in$  [ x, x + intValue ]
 $\forall$  cells [INDEX(x', y, z)]::pObject
    := prevObject, x'  $\in$  [ x, x + intValue ]
x = x + intValue
if floatValue more than intValue then
    x = x + 1
    W = (floatValue - intValue) / delta[1]
    cells [INDEX(x, y, z)]::Coef[X]
        := EffectiveMaterial( prevMaterial, IObject::MP[X], W)
    cells [INDEX(x, y, z)]::BE := XN | YN | ZN
    if prevObject is NULL then
        cells [INDEX(x, y, z)]::pObject := IObject
    else
        cells [INDEX(x, y, z)]::pObject := prevObject
    end if
end if
prevMaterial := IObject::M[X]
else
     $\forall$  cells [INDEX(x', y, z)] ::Coef[X]
        := material of free space, x'  $\in$  [ x, bounds[2][X] ]
    cells [INDEX(x + 1, y, z)]::BE := XN | YN | ZN
end if
end while
end while
//Shoot ray to +Y direction at (X, bounds[1][Y], Z)
y = bounds[1][Y]; Ray::Direction := (0, 1, 0); prevMaterial = material of
free space
-- skip --
prevObject := Object in which ray start.
if FindIntersection(rootKD, Ray, iP, IObject) then

```

```

floatValue = (iP[Y] - x) / delta[Y]
intValue = int(floatValue)
 $\forall$  cells [INDEX(x, y', z)]::Coef[Y]
    := prevMaterial, y'  $\in$  [ y, y + intValue ]
if cells [INDEX(x, y', z)]::pObject is NULL then
    cells [INDEX(x, y', z)]::pObject := prevObject
end if
x = x + intValue
if floatValue more than intValue then
    x = x + 1
    W = (floatValue - intValue) / delta[1]
    cells [INDEX(x, y, z)]::Coef[X]
        := EffectiveMaterial( prevMaterial, IObject::MP[X], W)
    cells [INDEX(x, y, z)]::BE := XN | YN | ZN
    if cells [INDEX(x, y, z)]::pObject is NULL then
        if prevObject is NULL then
            cells [INDEX(x, y, z)]::pObject := IObject
        else
            cells [INDEX(x, y, z)]::pObject := prevObject
        end if
    end if
end if
prevMaterial := IObject::M[Y]
else
     $\forall$  cells [INDEX(x, y', z)] ::Coef[X]
        := material of free space, y'  $\in$  [ y, bounds[2][Y] ]
    cells [INDEX(x, y+1, z)]::BE := XN | YN | ZN
end if
-- skip --
//Shoot ray to +Z direction at (X, Z, bounds[1][Z])
z = bounds[1][Z]; Ray::Direction := (0, 0, 1); prevMaterial = material of
free space

```

**\*\* Similar to the Y case**

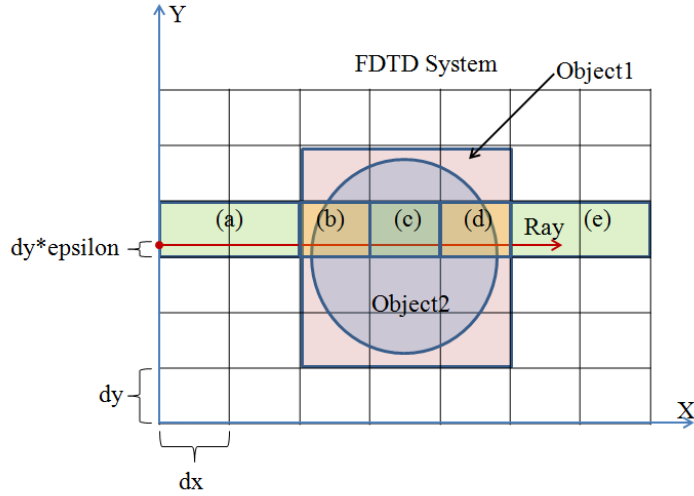
**end function**

---

One simple example illustrated in Figure 57 was prepared. In this case, the algorithm discretize the system into the X direction, and subsequently, set the boundary and object information of the discretized cells. If the ray intersect with 'Object1' at the first time then the (a) cells are determined with that the cells' 'BE' value is unchanged and 'pObject' is set with NULL because they exist in the free space and the intersection point exists exactly on the edge of the cell. 'prevObject' was set with the reference value of 'Object1' before 'FindIntersection' function is called.

The second intersection occurs inside the (b) cell with 'Object2'. At that time, the (b) cell is set with the suitable values. According to Algorithm 20, the (b) cell's 'BE' is set with XN, YN, and ZN flags, and the cell's 'pObject' is set with 'prevObject' (*i.e.*, the reference value of the Object1). 'prevObject' was set with the reference value of 'Object1' because the ray's starting position existed inside 'Object1'.

The third intersection occurs inside the (d) cell with 'Object2'. The (c) cell's 'BE' is unchanged, and its 'pObject' is set with the reference value of 'Object2'. The (d) cell's 'BE' is set with XN, YN, and ZN flags, and its 'pObject' is set with the reference value of 'Object2'. 'prevObject' was set with 'Object 2'. The (e) cells keep their initial values because there is no more intersection. As a result, Algorithm 20 assigned the (b) and (c) cells as boundary cells at the end of this operation.



**Figure 57.** Schematic of the discretization of immersed two objects in the FDTD system.

Algorithm 20 found the objects' boundary and saved their information at the discretized cells. However, these boundary cells make a discontinuous and opened boundary surface. Thus, these have to be converted to the continuous and closed one because the divergence theorem needs a continuous and closed surface. To solve this conversion problem, I developed 'MakeContinuous-Boundary' function that is described in Algorithm 21.

---

**Algorithm 21.** Make Continuous Boundary

---

**function** MakeContinuousBoundary (Cell cells[])

**while** x from 1 to # of X **do**

**while** y from 1 to # of Y **do**

**while** z from 1 to # of Z **do**

                //Set -X and +X normal surface boundary

                prevCell := cells [x-1, y, z]

                curCell := cells [x, y, z]

**if** prevCell::BE has XP **then**

**if** curCell::pObject equals prevCell::pObject **then**

                        Unset XP flag from prevCell::BE



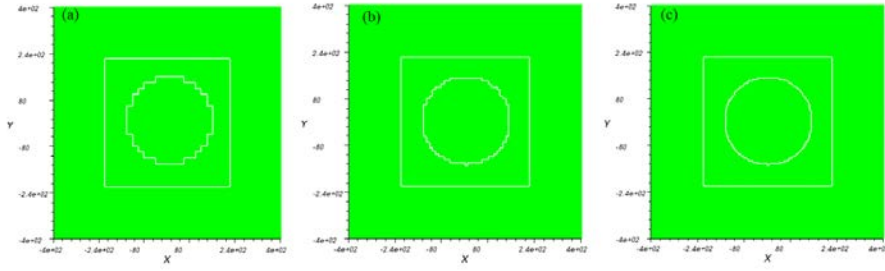
```

    else
        Set XN flag at curCell::BE
    end if
end if
if curCell::BE has boundary flag then
    if curCell::pObject equals prevCell::pObject then
        Unset XN flag from curCell::BE
    else
        Add XP flag to the curCell::BE
    end if
    if x is greater than second cell and curCell::BE has XN flag and
        curCell::pObject not equal prevCell::pObject then
        Add XP flag to the curCell::BE
    end if
end if
//Set -Y and +Y normal surface boundary
Similar to the -X and +X normal surface boundary logic
//Set -Z and +Z normal surface boundary
Similar to the -X and +X normal surface boundary logic
end while
end while
end while
end function

```

---

Algorithm 21 is a kind of sewing algorithm which set or unset the boundary flags of the cells made by Algorithm 20 according to a condition of their vicinity cells. With Algorithm 21, all the boundary cells are converted into a continuous and closed boundary surface.

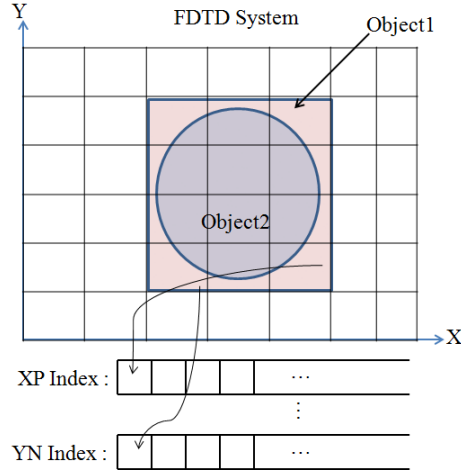


**Figure 58.** Illustration of the results of the algorithm 14 with the FDTD system of Fig. 55. (a) is 20nm cell size, (b) is 10nm cell size, and (c) is 5nm cell size.

Figure 58 is an example of the results of Algorithm 21. In this figure, it can be found that the three boundaries keep their continuity independent of their cell sizes, so the divergence part of the Poynting theorem can be used in the calculation of the absorption energy.

#### 4.5.1.2 Memory allocation and index mapping for the boundary cells

The next procedure is allocating of memory for the boundary cells and mapping the index of the boundary cells. Through this process, I could reduce the required memory and the computational time for the Discrete Fourier Transform(DFT) operation and surface integration. The process make six (or four in 2-D) index lists, which contains the index of the boundary cells according to the surface flags like Figure 59, and twenty-four (or sixteen in 2-D) field value lists which contains the DFT values for the two transversal E- and H-fields pairs, for each of the objects of which I want to know the absorption energy and for each spectral ranges. These E- and H-fields values are used to calculate the Poynting power at the each inward surface normal directions.



**Figure 59.** Schematics of the extraction of the boundary cell indices.

The DFT calculation is processed at the last of the each FDTD time-steps with the equation

$$F_{DFT} = F_{DFT} + F_{\text{current value}} \times e^{-\omega \cdot n \cdot \Delta t}, \quad (4.10)$$

where  $F$  is E- or H-field,  $\omega$  is an angular frequency,  $n$  is the number of current time step, and  $\Delta t$  is the time interval for the FDTD system.

#### 4.5.1.3 Calculation of the absorption energy

At the last of the FDTD time step, I calculated the absorption energy of all the objects inside the FDTD system for all spectral ranges with the right hand side of Equation(4.9). The Poynting power was calculated with the DFT results of the transversal E- and H-fields pairs for all frequencies, for all objects and for all surfaces because the boundary surfaces were made by Yee's cell and the normal vectors had only six vector values (*i.e.*, -X, +X, -Y, +Y, -Z, +Z).

Thus, the surface integration for the divergence calculation is a simple summation of the multiplication values of the each bounding cell's Poynting power by  $\pm 1.0$  according to the cell's boundary flags for all six surface types. This procedure was summarized in Algorithm 22.

---

**Algorithm 22.** Calculate Absorption Energy

---

```
function CalculateAbsorption ()  
  while all objects do  
    while all frequencies do  
      Energy := 0.0  
      while all number of XP list do  
        Energy = Energy -  $\{(E_y \times H_z^*) - (E_z \times H_y^*)\} \times dydz$   
      end while  
      while all number of XN list do  
        Energy = Energy +  $\{(E_y \times H_z^*) - (E_z \times H_y^*)\} \times dydz$   
      end while  
      YP, YN, ZP, and ZN similar to the XP and XN.  
    end while  
  end while  
  return Energy  
end function
```

---

where \* means complex conjugate value and 'dydz' is a cell area of XP surface.

However, This used a simple discrete integration method, so it had order of one convergence. To improve order of convergence, I applied Simpson rule to the integration part of Algorithm 22. Before apply Simpson rule, I had to consider the topology information of the boundary cells, because the boundary surfaces in the XP, XN, YP, YN, ZP, and ZN list were not located on the plane surface. That topology information was saved in the XP, XN, YP, YN, ZP, and ZN Index list, so new algorithm found the location of the boundary surfaces using these Index list and executed Simpson rule. The improved algorithm like this :

---

**Algorithm 23.** Improved Calculation of Absorption Energy

---

```
function CalculateAbsorption ()  
  while all objects do  
    while all frequencies do  
      Energy := 0.0
```

```

while all number of XP list do
    int j := XP_Index[Y]
    int k := XP_Index[Z]
    float W := 0.0
    if (j == 1 || j == # of Y cell) && (k == 1 || k == # of Z cell) then
        W := 1.0;
    else if (j == 1 && k%2 == 1) || (j == # of Y cell && k%2 == 1) ||
        (j%2 == 1 && k == 0) || (j%2 == 1 && k == # of Z cell) then
        W := 2.0
    else if (j == 1 && k%2 == 0) || (j == # of Y cell && k%2 == 0) ||
        (j%2 == 0 && k == 0) || (j%2 == 0 && k == # of Z cell) || (j%2
        == 1 && k%2 == 1) then
        W := 4.0
    else if (j%2 == 0 && k%2 == 1) || (j%2 == 1 && k%2 == 0) then
        W := 8.0
    else
        W := 16
    end if
    Energy = Energy - {(Ey × Hz*) - (Ez × Hy*)} × dydz × W / 9.0
end while
    XN, YP, YN, ZP, and ZN similar to the XP.
end while
end while
return Energy
end function

```

---

where \* means complex conjugate value and 'dydz' is a cell area of XP surface.

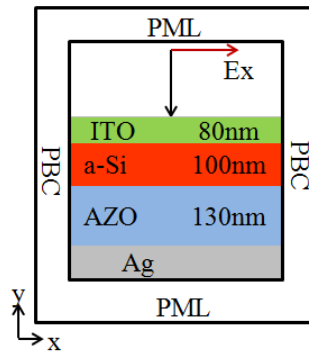
#### 4.5.2 Numerical Analysis

In this section, I validate Algorithm 23 using the ray tracing algorithm of Chapter 3 and other reference literatures with flat layer model, and also analyzed an absorption energy of a non-flat active layer model.

#### 4.5.2.1 Flat system.

To validate Algorithm 23, I prepared a normal thin film amorphous silicon solar cell model which was consisted of all flat layers. This model was composed of 80nm thick Indium Tin Oxide (ITO) anti-reflection coating, 100nm thick amorphous silicon (a-Si), 130nm thick Al doped zinc oxide (AZO) back-contact, and sufficiently thick silver (Ag) back-reflector [110]. The 2-D FDTD layout of this model was depicted in Figure 60. This FDTD system was discretized with an even 1nm subdivision delta value in the X and Y directions by the auto-discretization algorithm introduced in Section 4.2.

To simulate real solar cell model with this FDTD system, the incident light source was set an  $E_x$  polarized normal incident Gaussian pulse with 200 - 900nm wavelength, the left and right sides of the system were set periodic boundary condition (PBC) these enable to simulate the infinite length system in X direction, and the top and bottom were set PML absorbing boundary condition to ensure that a penetrated energy into the Ag layer is entirely absorbed by the Ag layer.



**Figure 60.** Layout for the 2-D FDTD model of the flat thin film amorphous silicon solar cell.

Also, the ITO and AZO material property need to be set. These two materials have dispersive material property under the 200 - 900nm wavelength range. Thus, I had to prepare the Lorentz parameters for these materials. The fitted Lorentz parameters of the ITO and the AZO were

(a)			
Pole No.	Delta Epsilon	Plasma Frequency	Damping Frequency
1	2.781140E+00	1.179170E+16	5.893650E+08
2	3.068520E-01	7.666790E+15	1.287110E+15
3	7.994200E-02	2.872750E+15	1.158720E+15
4	2.719563E+01	5.012550E+14	1.246230E+14

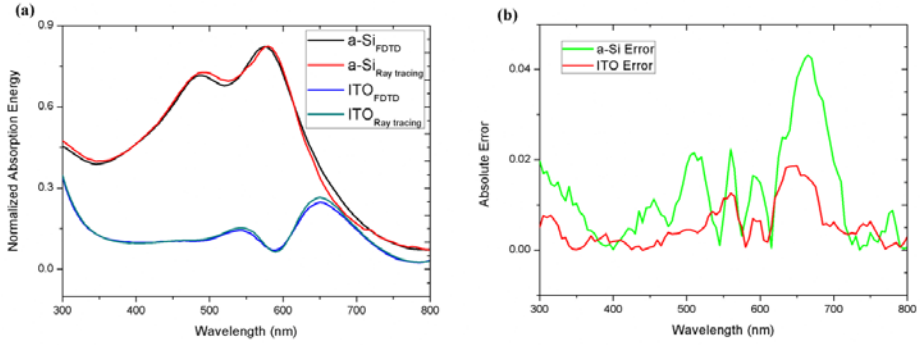
(b)			
Pole No.	Delta Epsilon	Plasma Frequency	Damping Frequency
1	1.727885E+00	9.028580E+15	2.994180E+14
2	-1.181783E+00	5.666870E+15	1.528200E+15
3	1.390847E+00	6.537720E+15	1.186290E+15
4	2.703960E-01	5.051850E+15	1.341690E+15

Table 5. (a) Lorentz parameters for ITO and (b) for AZO.

where epsilon infinity( $\epsilon_{\infty}$ ) was 1.0, the experiment data of ITO and AZO was used in the SOPRA nk database [102], and the Lorentz parameters for Ag was the same values in [74].

To validate the algorithm, I had to prepare reference data for the error analysis. For this purpose, the ray tracing algorithm of Chapter 3 was used for calculating the absorption energy of the each layer because this model has all flat layers and plasmon effect is not dominant in this case. Therefore, I simulated the model with the two kinds of methods. The one is FDTD method with the previous algorithm, which simulated with 80,000 time steps in 2-D TE mode, and the other is the ray tracing method with the algorithm of Chapter 3 which simulated with total 10,000,000 rays.

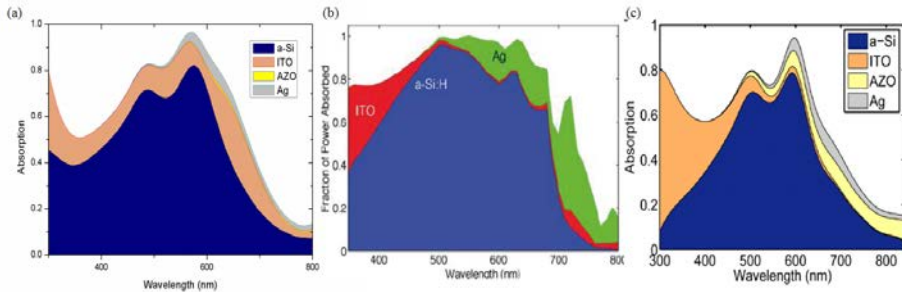
The calculated absorption energies of the a-Si and the ITO layers were compared with each other. The comparison results were reported in Figure 61 with the result of the absolute error values.



**Figure 61.** (a) Comparison results of the FDTD and Ray-tracing, (b) Absolute error values.

The maximum error of the a-Si was less than 5%, and the maximum error of the ITO was less than 2%. From this results, the correctness of my algorithm could be validated. In this model, I did not suffer the staircase error problem because the 1nm cell size was fitted to the layers' boundary exactly. Instead, I suffered another intrinsic error derived from Yee's scheme. The central difference method calculates an E-field value with the difference values of H-fields around it. Therefore, the calculated E-field value was an averaged value across the objects' boundary, and this fact cause a simulation error.

For more comparison, I prepared three kinds of stacked graph in Figure 62, (a) came from my FDTD model, (b) came from Ferry, et al. [113], and (c) came from Grandidier, et al. [110].



**Figure 62.** (a) Results of the FDTD, (b) Image from [113], (c) Image from [110].

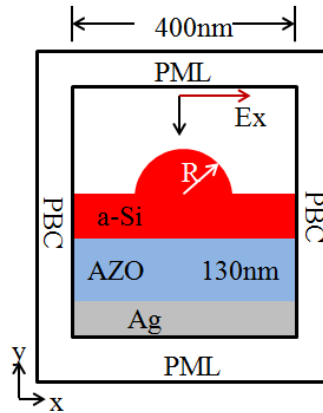
Figure 62(b) is the spectral absorption data from 350nm to 800nm wavelength for Asahi glass embedded thin film solar cell. This model has the similar ab-



sorption energy with the flat model less than 400nm wavelength. Figure 62(c) is the spectral absorption data from 300nm to 800nm wavelength for the flat thin film solar cell which is same to my model. At the 350nm wavelength, Figure 62(a) and Figure 62(b) have similar absorption energy for a-Si. Meanwhile, for the total absorption energy in the entire range, Figure 62(a) and Figure 62(c) is similar to each other. This means that my algorithm creates more accurate results than Figure 62(b) and Figure 62(c).

#### 4.5.2.2 Non-Flat system.

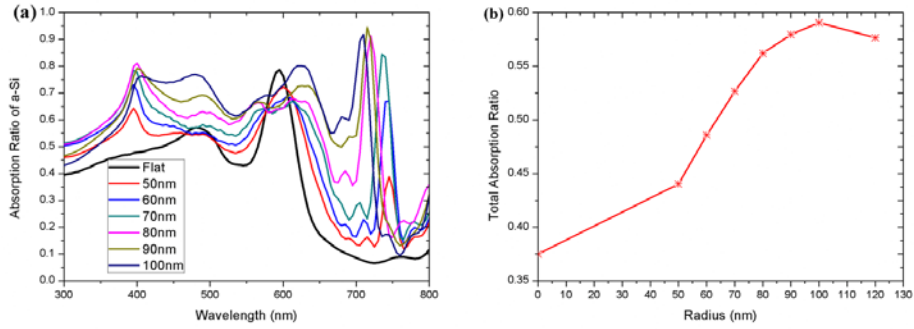
In this section, I slightly changed the previous section's model to simulate a non-flat active layer. I removed the ITO layer, and changed the flat a-Si layer to a hemisphere embedded a-Si layer. The other conditions were not changed as Figure 63.



**Figure 63.** Layout for the 2-D FDTD model of the non-flat thin film amorphous silicon solar cell.

The radius of the hemisphere was increased by 10nm start from 50nm to 120nm keeping the total volume of the a-Si layer. The FDTD system was discretized with an even 2.5nm space delta in all X and Y axes, and the input source was total-field/scattered-field (TF/SF) mode single Gaussian pulse with 200 - 900nm wavelength. All models were simulated at  $TE_z$  and  $TM_z$  mode, and these two values were averaged. With these simulation results, I analyzed the enhanced absorption energies according to the various radii, and also compared the absorption energies to the flat a-Si layer's one and each

other. This model had antireflection effect with nanograting pattern and total internal reflection(TIR) effect with hemisphere both. Therefore, as the radius of hemisphere changes, the antireflection frequency and total internal reflection(TIR) frequency also changes. These all phenomena were found in my simulation results as Figure 64.



**Figure 64.** (a) Spectral comparison of absorption energy ratio, (b) Comparison of total absorption energy ratio.

Figure 64(a) revealed that the TIR frequency shifts to the higher frequency region as the radius of the hemisphere increased. Meanwhile, the antireflection frequency moved in the opposite direction of the TIR frequency. These two effects enhanced the absorption efficiency. Figure 64(b) showed that the total absorption energy increased as the radius increases until the 100nm. This was a simple numerical example to illustrate the accuracy of my algorithm. It also can be used in arbitrary shaped geometries.

Through these two kinds of numerical models, I validated my algorithm and showed the efficiency of finding the optimal structure to achieve maximum absorption energy enhancement in the thin film silicon solar cells.

## 4.6 Conclusion

In this chapter, I proposed several FDTD algorithms such as the fast and robust auto-discretized algorithm, two kinds of effective material algorithms (*i.e.*, Round-off, D-CFDTD), and the continuous boundary extraction algorithm.

All these algorithms were aim for precisely simulating the absorption energy of the arbitrary shaped object embedded thin film solar cells with a direct method. In my knowledge, this was the first implementation of the divergence theorem for the arbitrary shaped objects. All these algorithms were validated by suitable methods, respectively, and also I showed several numerical analysis models using these algorithms.

With these algorithms, I achieved my research goal to analyze the absorption energy of the sub-wavelength scale thin film solar cells directly with the correct and efficient FDTD method. Also, all these algorithms were applied to the commercial FDTD tool (*i.e.*, Raywiz-FDTD) [117].

## 5 Conclusion

### 5.1 Summary

In this thesis, I proposed several numerical methods to analyze the enhanced absorption energy of the thin film silicon solar cells directly. The Slab-Outline algorithm is a novel algorithm which enables to find an intersection point for an enormous number of particles embedded object with asymptotically average  $O(N_r \times N_i \times \log 2N_o)$  time complexity. With this algorithm, the enhanced absorption energy of the pyramid textured thin film solar cells can be simulated efficiently and accurately. The direct absorption energy calculation algorithm for the ray tracing method was also proposed in chapter 3. In chapter 4, I proposed the auto-discretization algorithm, Round-Off algorithm, D-CFDTD algorithm, and continuous boundary extraction algorithm for the FDTD method.

### 5.2 Evaluation

When I started my research, I want to achieve an efficient and an accurate numerical method to calculate the enhanced efficiency of the thin film silicon solar cells. The importance of these numerical methods explained in the introductory section.

To reach this research goal, I divided my problem into the three kinds of sub-problems according to the system size. Such that, the first was geometric optics area, the second was wave optics area, and the third was electromagnetic optics area. These all areas can be simulated with the FDTD method. However, if the electromagnetic optics is applied to the large scale(*i.e.*,  $\gg$  optical wavelength( $\lambda$ )) problems, it requires massive computing time. Therefore, geometric and wave optics are needed for the large and middle scale(*i.e.*,  $\approx$  optical wavelength( $\lambda$ )) problems.

To solve the geometric optics system, many researchers use the ray tracing method. However, if this system has an enormous number of particles embedded object, the required ray tracing time increased linearly or quadratically

according to the number of particles. This computing time was reduced by the Slab-Outline algorithm to the  $O(\log N)$  time.

In Chapter 3, I proposed the direct calculation algorithm of the absorption energy for the micrometer and nanometer scale thin film solar cells. In the past, the direct calculation of the absorption energy for the geometric and wave optics system was impossible within my knowledge. Also, I extended the utility of the ray tracing method to the wave optics system with my algorithm proposed in Chapter 3.

The algorithms of Chapter 4 also enabled the absorption energy calculation of the sub-wavelength thin film solar cells with FDTD method accurately and efficiently.

### **5.3 Future Work**

In the FDTD method, there are many research issues. Especially, in the dispersive material part, I used relatively many pole-pair Lorentz parameters to achieve accurate result. However, this requested more computational time, so it is needed to reduce the number of parameters with keeping their accuracy. Next, I have to solve the staircase problem which occurs when a curved metallic dispersive material meets dielectric material. In this case, the surface plasmon effect is dominant, so I cannot use the D-CFDTD algorithm.

## References

- [1] Saleh & Teich, *Fundamentals of Photonics*, Second Edition. John Wiley Sons, 1991.
- [2] Rolf Brendel, *Thin-Film Crystalline Silicon Solar Cells*, First Edition. Willey-VCH, 2003.
- [3] Pharr & Humphreys, *Physically Based Rendering*, First Edition. Elsevier Inc., 2004.
- [4] Henrik Wann Jensen, *Realistic Image Synthesis Using Photon Mapping*, First Edition. A K Peters, Ltd., 2001.
- [5] I. Wald and V. Havran. *On Building Fast KD-Trees for Ray Tracing, and on doing that in  $O(N\log N)$* . In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, pp. 61-70, 2006.
- [6] V. Havran and J. Bittner. *On Improving KD-Trees for Ray Tracing*. Journal of WSCG, 10(1):209-216, 2002.
- [7] W. Hunt and W. R. Mark. *Fast KD-Tree Construction with an Adaptive Error-Bounded Heuristic*. IEEE Symposium on Interactive Ray Tracing 2006, pp. 18-20, 2006.
- [8] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral and K. Zikan. *Efficient Collision Detection Using Bounding Volume Hierarchies of  $k$ -DOPs*. IEEE Transaction on Visualization and Computer Graphics, Vol. 4, No. 1, Jan.-Mar. 1998.
- [9] T. Larsson and T. Akenine-Moller. *A dynamic bounding volume hierarchy for generalized collision detection*. Computer & Graphics, Vol. 30, Issue 3, pp. 450-459, Jun. 2006.
- [10] T. L. Kay and J. T. Kajiya. *Ray tracing complex surfaces*. Computer Graphics, 20(4):269-278, Aug. 1986.
- [11] H. Weghorst, G. Hooper and D. P. Greenberg. *Improved Computational Methods for Ray Tracing*. ACM Transactions on Graphics, Vol. 3, Issue 1, pp. 52-69, Jan. 1984.
- [12] S. Y. Byun, S.-J. Byun, J. K. Lee, J. W. Kim, T. S. Lee, D. Sheen, K. Cho, S. J. Park, D. Kim and W. M. Kim, *An efficient ray tracing algorithm for*

- the simulation of light trapping effects in Si solar cells with textured surfaces*. J. Nanosci. Nanotech., Vol. 12, pp. 3224-3227, 2012.
- [13] Allen Taflove and Susan C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Second Edition. Artech House, Inc., 2000.
- [14] K. S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*. IEEE Trans. Antennas and Propagation, Vol. 14, pp. 302-307, 1996.
- [15] J. Haynos, J. Allison, R. Arndt, and A. Meulenbergh, *The comsat nonreflective silicon solar cell : A second generation improved cell*, Int. Conf. Photovoltaic Power Generation, Hamburg , p. 487, 1974.
- [16] P. Campbell and M. A. Green, *Light trapping properties of pyramidally textured surfaces*, J. Appl. Phys. 62, p.243, 1987 .
- [17] J. Poortmans, J. Nus, and R. Mertens, Clean Electricity from Photovoltaics, edited by M. D. Archer and R. Hill, Imperial College Press, London , Vol. 1, p. 99, 2001.
- [18] A. W. Smith, A. Rhoatgi, and S. C. Neel, *Texture: A ray tracing program for the photovoltaic community*, Proc. 21st IEEE Photovoltaic Specialist Conf., p.426, 1990.
- [19] P. Campbell, S. R. Wenham, and M. A. Green, *Light trapping and reflection control in solar cells using tilted crystallographic surface textures*, Sol. Energy Mater Sol. Cells 31, pp.133-153, 1993.
- [20] R. Brendel, *Coupling of Light into Mechanically Textured Silicon Solar Cells*, Prog. Photovolt. 3, pp.25-38, 1995.
- [21] J. M. Gee, W. K. Schubert, H. L. Tardy, and T. D. Hund, *The effect of encapsulation on the reflectance of photovoltaic modules using textured multicrystalline-silicon solar cells*, 1st World Conf. Photovoltaic Energy Conversion, Waikoloa, Hawaii, pp.1555-1558, 1994.
- [22] S. Yoon, G. Turner, and V. Garboushian, *Development of back junction point contact photovoltaic cells and arrays for space*, Proc. 25th PVSC, Washington, D.C., p. 259, 1996.

- [23] T. Yagi, Y. Uraoka, and T. Fuyuki, *Ray-trace simulation of light trapping in silicon solar cell with texture structures*, Sol. Energy Mater. Sol. Cells 90, pp.2647-2656, 2006.
- [24] J. M. Rodriguez, I. Tobias, and A. Luque, *Random pyramidal texture modelling*, Sol. Energy Mater. Sol. Cells 45, pp.241-253, 1997.
- [25] D. Thorp and S. R. Wenham, *Ray-tracing of arbitrary surface textures for light-trapping in thin silicon solar cells*, Sol. Energy Mater. Sol. Cells 48, pp.295-301, 1997.
- [26] K. R. McIntosh, R. M. Swanson, and J. E. Cotter, *A Simple Ray Tracer to Compute the optical concentration of photovoltaic modules*, Prog. Photovolt. : Res. Appl. 14, 167 2006.
- [27] InsideOptics Co., Ltd., Raywiz-SOLAR User Manual 2010.
- [28] J. L. Bentley, *Multidimensional binary search trees used for associative searching*, Comm. ACM 18, pp.509-517, 1975.
- [29] S.-J. Byun, S. Y. Byun, J. Lee, J. W. Kim, T. S. Lee, W. M. Kim, Y. K. Park, and K. Cho, *An optical simulation algorithm based on ray tracing technique for light absorption in thin film solar cells*, Sol. Energy Mater. Sol. Cells 95, pp.408-411, 2011.
- [30] M. Gervautz, *Three Improvements of the ray tracing algorithm for CSG trees*, Comput. Graph. 10, pp.333-339, 1986.
- [31] M.A. Green, J. Zhao, A. Wang, S.R. Wenham, *Very High Efficiency Silicon Solar Cells-Science and Technology*, IEEE. Trans. on Electron Dev. Vol. 46, 10, pp.1940-1947, 1999.
- [32] J.D. Hylton, *Light coupling and light trapping in alkaline etched multicrystalline silicon wafers for solar cells*, Ph. D. Thesis, Utrecht University, Netherlands, p.74, 2006.
- [33] B. Prasad, S. Bhattacharya, A.K. Saxena, S.R. Reddy, R.K. Bhogra, *Performance enhancement of mc-Si solar cells due to synergetic effect of plasma texturization and SiNx:H AR coating*, Sol. Energy Mater. Sol. Cells 94, pp.1329-1332, 2010.



- [34] J. D. Hylton, A. Burgers, and W.C. Sinke, *Absorption in thin textured silicon wafers*, in: Proceedings of the 14th European Photovoltaic Solar Energy Conference, pp. 139–142, 1997.
- [35] A. Mavrokefalos, S.E. Han, S. Yerci, M.S. Branham, G. Chen, *Efficient Light Trapping in Inverted Nanopyramid Thin Crystalline Silicon Membranes for Solar Cell Applications*, Nano Lett., Vol. 12, pp. 2792-2796, 2012.
- [36] Requicha, A. A. G. and H. B. Voelcker, *Constructive Solid Geometry*, Production Automation Project Technical Memorandum TM-25, April 1980.
- [37] S. D. Roth, *Ray casting for modeling solid*, Computer Graphics Image Process, 18, pp. 109-144, 1980.
- [38] C. Wachter, A. Keller, *Instant Ray Tracing: The Bounding Interval Hierarchy*, In *Rendering Techniques 2006 – Proceedings of the 17th Eurographics Symposium on Rendering* (2006), pp. 139-149, 2006.
- [39] R. Dewan, D. Knipp, *Light trapping in thin-film silicon solar cells with integrated diffraction grating*, J. Applied Physics 106, 074901, 2009.
- [40] J.M. Snyder, A.H. Barr, *Ray Tracing Complex Models Containing Surface Tessellations*, ACM Computer Graphics, Vol. 21, No. 4, pp. 119-128, July, 1987.
- [41] S.M. Rubin, T. Whitted, *A 3-Dimensional Representation for Fast Rendering of Complex Scenes*, ACM SIGGRAPH Computer Graphics, Vol. 14, No. 3, pp. 110-116, July, 1980.
- [42] J. D. MacDonald, K. S. Booth, *Heuristics for Ray Tracing using Space Subdivision*, Visual Comput., Vol. 6, pp. 153-166, 1990.
- [43] J. H. Friedman, J. L. Bentley, R. A. Finkel, *An Algorithm for finding best matches in logarithmic time*, ACM Tran. on Math. Software, Vol. 3, No. 3, pp. 209-226, 1977.
- [44] S. -J. Byun, S. Y. Byun, J. K. Lee, J. W. Kim, T. S. Lee, K. M. Cho. D. Sheen, S. J. Tark, D. H. Kim, W. M. Kim, *Analysis of light trapping effects in Si solar cells with a textured surface by ray tracing simulation*, Current Applied Physics, pp. 1-3, 2011.

- [45] M. Muñoz, P. Carreras, J. Escarré, D. Ibarz, S. Martín de Nicolás, C. Voz, J. M. Asensi, J. Bertomeu, *Optimization of KOH etching process to obtain textured substrates suitable for heterojunction solar cells fabricated by HWCVD*, Thin Solid Films, Vol. 517, pp. 3578-3580, 2009.
- [46] Y. Srisukh, J. Nehrbass, F. L. Teixeira, F.-F. Lee, R. Lee, *An approach for automatic grid generation in three-dimensional FDTD simulations of complex geometris*, IEEE Antennas and Propagation Magazine, Vol. 44, Issue: 4, pp. 75-80, 2002.
- [47] T. Su, Y. Liu, W. Yu and R. Mittra, *A Conformal Mesh-Generating Technique for the Conformal Finite-Difference Time-Domain (CFDTD) Method*, IEEE Antennas Propagat.Mag., Vol.46, No.1, pp.37-49, 2004.
- [48] R. Holland, *Finite-difference solution of Maxwell's equations in generalized nonorthogonal coordinates*, IEEE Trans. Nucl. Sci., 30, pp. 4589-4591, 1983
- [49] K. K. Mei, A. Cangellaris, and D. J. Angelakos, *Conformal time domain finite difference method*, Radio Sri., vol. 19, pp. 1145-1147, 1984.
- [50] M.A. Fusco, M.V. Smith, L.W. Gordon, *A three-dimensional FDTD algorithm in curvilinear coordinates*, IEEE Trans Antennas Propag, 39, pp. 1463-1471, 1991.
- [51] P. H. Harms, J. F. Lee, and R. Mittra, *A study of the nonorthogonal FDTD method versus the conventional FDTD technique for computing resonant frequencies of cylindrical cavities*, IEEE Trans. Microwave Theory Tech., vol. 40, pp. 741-746, 1992.
- [52] T. G. Jurgens, A. Taflove, K. Umashankar, T. G. Moore, *Finite difference time-domain modeling of curved surfaces*, IEEE Trans. Antennas Propagat., vol. 40, pp. 357-365, 1992.
- [53] T. G. Jurgens, A. Taflove, *Three-dimensional contour FDTD modeling of scattering from single and multiple bodies*, IEEE Trans. Antennas Propagat., vol. 41, pp. 1703-1708, 1993.
- [54] Y. Hao, C.J. Railton, *Analyzing electromagnetic structures with curved boundaries on Cartesian FDTD meshes*, IEEE Trans. Antennas Propagat., vol. 46, pp. 82-88, 1998.

- [55] I. S. Kim and W. J. R. Hoefer, *A local mesh refinement for the time-domain finite-difference method using Maxwell's curl equations*, IEEE Trans. Microwave Theory Tech., vol. 38, pp. 812–815, June 1990.
- [56] J.-P. Bérenger, *A Huygens Subgridding for the FDTD method*, IEEE Trans. Antennas Propagat., vol. 54, pp. 3797–3804, 2006.
- [57] M. W. Chevalier, R. J. Luebbers, V. P. Cable, *FDTD local grid with material traverse*, IEEE Trans. Antennas Propagat., vol. 45, pp. 411–421, 1997.
- [58] K. M. Krishnaiah and C. J. Railton, *Passive equivalent circuit of FDTD: An application to subgridding*, Electron. Lett., vol. 33, pp. 1277–1278, 1997.
- [59] L. Kulas and M. Mrozowski, *A simple high-accuracy subgridding scheme*, 33rd Eur. Microwave Conf., Munich, Germany, pp. 347–350, Oct. 2003.
- [60] L. Kulas and M. Mrozowski, *Low reflection subgridding*, IEEE Trans. Microwave Theory Tech., vol. 53, no. 5, pp. 1587–1592, May 2005.
- [61] M. Okoniewski, E. Okoniewski, M. A. Stuchly, *Three-dimensional subgridding algorithm for FDTD*, IEEE Trans. Antennas Propagat., vol. 45, pp. 422–429, Mar. 1997.
- [62] D. T. Prescott, N. V. Shuley, *A method for incorporating different sized cells into the finite-difference time-domain analysis technique*, IEEE Microwave Guided Wave Lett., vol. 2, pp. 434–436, Nov. 1992.
- [63] N. V. Venkatarayalu, R. Lee, Y.-B. Gan, L.-W. Li, *A stable FDTD subgridding method based on finite element formulation with hanging variables*, IEEE Trans. Antennas Propag., vol. 55, pp. 907–915, 2007.
- [64] M. J. White, Z. Yun, M. F. Iskander, *A new 3d FDTD multigrid technique with dielectric traverse capabilities*, IEEE Trans. Microw. Theory Tech., vol. 49, pp. 422–430, 2001.
- [65] K. Xiao, D. J. Pommerenke, J. L. Drewniak, *A three-dimensional FDTD subgridding algorithm with separated temporal and spatial interfaces and related stability analysis*, IEEE Trans. Antennas Propagat., vol. 55, pp. 1981–1990, 2007.

- [66] S. S. Zivanovic, K. S. Yee, K. K. Mei, *A subgridding method for the time-domain finite-difference method to solve Maxwell's equations*, IEEE Trans. Microw. Theory Tech., vol. 39, pp. 471–479, Mar. 1991.
- [67] N. Kaneda, B. Houshmand, T. Itoh, *FDTD Analysis of Dielectric Resonators with Curved Surfaces*, IEEE Trans. Microw. Theory Tech., vol. 45, pp. 1645–1649, 1997.
- [68] T. Hirono, Y. Shibata, W. W. Lui, S. Seki, Y. Yoshikuni, *The Second-order Condition for the Dielectric Interface Orthogonal to the Yee-lattice axis in the FDTD scheme*, IEEE Microwave Guided Wave Lett., vol. 10, pp. 359–361, 2000.
- [69] K.-P. Hwang, A. C. Cangellaris, *Effective Permittivities for Second-order Accurate FDTD equations at Dielectric Interfaces*, IEEE Microwave Wireless Components Lett., vol. 11, pp. 158–160, Apr. 2001.
- [70] S. Dey, R. Mittra, *A Conformal Finite-Difference Time-Domain Technique for Modeling Cylindrical Dielectric Resonators*, IEEE Trans. Microwave Theory Tech., vol. 47, pp. 1737–1739, 1999.
- [71] W. Yu, R. Mittra, *A Conformal Finite Difference Time Domain Technique for Modeling Curved Dielectric Surfaces*, IEEE Microwave Wireless Comp., vol. 11, pp. 25–27, 2001.
- [72] W. Yu, R. Mittra, *A Conformal FDTD Software Package Modeling Antennas and Microstrip Circuit Components*, IEEE Antennas Propagat., vol. 42, pp. 28–39, 2000.
- [73] A. D. Rakic, *Algorithm for the Determination of Intrinsic Optical Constants of Metal Films: application to aluminum*, Appl. Opt., vol. 34, pp. 4755–4767, 1995.
- [74] A. D. Rakic, A. Djurisiæ, J. Elazar, M. Majewski, *Optical properties of metallic films for vertical-cavity optoelectronic devices*, Appl. Opt., vol. 37, pp. 5271–5283, 1998.
- [75] Y. Zhao, Y. Hao, *Finite-Difference Time-Domain Study of Guided Modes in Nano-Plasmonic Waveguides*, IEEE Trans. Antennas Propagat., vol. 55, pp. 3070–3077, 2007.

- [76] N. Okada, J.B. Cole, *Effective Permittivity for FDTD Calculation of Plasmonic Materials*, Micromachines, vol. 3, pp. 168–179, 2012.
- [77] M. Okoniewski, M. Mrozowski, M. A. Stuchly, *Simple Treatment of Multi-Term Dispersion in FDTD*, IEEE Microwave Guided Wave Lett., vol. 7, No. 5, pp. 121–123, 1997.
- [78] J. A. Roden, S. D. Gedney, *Convolutional PML (CPML): An Efficient FDTD implementation of the CFS-PML for arbitrary media*, Micro. Opt. Technol. Lett., vol. 27, pp. 334–339, 2000.
- [79] J. Krč, M. Zeman, F. Smole, M. Topič, *Optical modeling of a a-Si:H solar cells deposited on textured glass/SnO<sub>2</sub> substrates*, J. Appl. Phys., vol. 92, pp. 749–755, 2002.
- [80] J. Krč, F. Smole, M. Topič, *Analysis of light scattering in amorphous Si:H solar cells by a one-dimensional semi-coherent optical model*, Prog. Photovolt: Res. Appl., vol. 11, pp. 15–26, 2003.
- [81] Yeh Pochi, in: *Optical Waves in Layered Media*, John Wiley & Sons, 1988.
- [82] InsideOptics Co., Ltd. RayWiz User Manual 2009.
- [83] Eugene Hecht, in: *Optics*, fourth ed., Addison Wesley, 2002.
- [84] J. Krč, M. Zeman, O. Kluth, F. Smole, M. Topič, *Effect of surface roughness of ZnO:Al films on light scattering in hydrogenated amorphous silicon solar cells*, Thin Solid Films, vol. 426, pp. 296–304, 2003.
- [85] S. -J. Byun, S. Y. Byun, T.S. Lee, W.M. Kim, K. Cho, D. Sheen, S.J. Tark, D. Kim, *High-efficiency Grid-type Si Solar Cell Structure*, J. Kor. Phys. Soc., vol. 60, pp. 2075–2078, 2012.
- [86] M. A. Green, *Clean Electricity from Photovoltaics*, edited by M.D. Archer and R. Hill (Imperial College Press, London 2001), vol. 1, p. 163.
- [87] B. Yan, G. Yue, L. Sivec, J. Owens-Mawson, J. Yang, S. Guha, *Correlation of texture of Ag/Zno back reflector and photocurrent in hydrogenated nanocrystalline silicon solar cells*, Sol. Energy Matter. Sol. cells, vol. 104, pp. 13-17, 2012.
- [88] S. J. Jones, D. Tsu, T. Liu, J. Steele, R. Capangpangan, M. Izu, *Development of transparent conductive oxide materials for improved back*

- reflector performance for amorphous silicon based solar cells*, Matter. Res. Soc. Symp. Proc., 808, A9.44.1-6, 2004.
- [89] A. Banerjee, S. Guha, *Study of back reflectors for amorphous silicon alloy solar cell application*, J. Appl. Phys., vol. 69, pp. 1030-1035, 1991.
- [90] J. Springer, B. Rech, W. Reetz, J.M. Uller, M. Vanecek, *Light trapping and optical losses in microcrystalline silicon pin solar cells deposited on surface-textured glass/ZnO substrates*, Sol. Energy. Matter. Sol. Cells, vol. 85, pp. 1-11, 2005.
- [91] X. D. Zhang, Y. Zhao, Y. T. Gao, F. Zhu, C. C. Wei, X.L. Chen, J. Sun, G. F. Hou, X. H. Geng, S. Z. Xiong, *Influence of front electrode and back reflector electrode on the performances of microcrystalline silicon solar cells*, J. Non-Cryst. Solids, vol. 352, pp. 1863-1867, 2006.
- [92] G. M. Nam, M. S. Kwon, *F-doped ZnO by sol-gel spin-coating as a transparent conducting thin film*, Electron. Mater. Lett., vol. 7, pp. 121-131, 2011.
- [93] Y. H. Kim, J. Jeong, K. S. Lee, B. Cheong, T.-Y. Seong, W. M. Kim, *Effect of composition and deposition temperature on the characteristics of Ga doped ZnO thin films*, Appl. Surf. Sci., vol. 257, pp. 109-115, 2010.
- [94] R.V. Muniswami Naidu, A. Subrahmanyam, A. Verger, M.K. Jain, S.V.N. Bhaskara Rao, S.N. Jha, D.M. Phase, *Electron-electron interactions based metal-insulator transition in Ga doped ZnO thin films*, Electron. Mater. Lett., vol. 8, pp. 457-462, 2012.
- [95] Z.-C. Jin, I. Hamberg, C. G. Granqvist, *Optical properties of sputter-deposited ZnO: Al thin films*, J. Appl. Phys., vol. 64, pp. 5117-5131, 1988.
- [96] H. Fujiwara, M. Kondo, *Effects of carrier concentration on the dielectric function of ZnO: Ga and In<sub>2</sub>O<sub>3</sub>:Sn studied by spectroscopic ellipsometry: analysis of free-carrier and band-edge absorption*, Phys. Rev. B 71 (2005) 075109.
- [97] E. D. Palik (Ed.), *Hand Book of Optical Constant of Solids*, vol. 1, Academic Press, San Diego 1998, ps. 350,369,547.

- [98] J. S. Kim, J.-h. Jeong, J. K. Park, Y. J. Baik, I. H. Kim, T.-Y. Seong, W. M. Kim, *Optical analysis of doped ZnO thin films using nonparabolic conduction-band parameters*, J. Appl. Phys. 111 (2012) 123507 .
- [99] S. Y. Byun, S.-J. Byun, T. S. Lee, D. Sheen, J.-H. Jeong, W. M. Kim, *Effect of oxide thin films in back contact on the optical absorption efficiency of thin crystalline Si solar cells*, Current Applied Physics (2013), <http://dx.doi.org/10.1016/j.cap.2013.01.045>.
- [100] D.W. Marquardt, *An Algorithm for Least-Squares estimation of nonlinear parameters*, J. Soc. Indust. Appl. Math., vol. 11, pp. 431-441, 1963.
- [101] A. Deinega, S. John, *Effective optical response of silicon to sunlight in the finite-difference time-domain method*, Opt. Lett., vol. 37, pp. 112-114, 2012.
- [102] SOPRA, nk database, <http://www.sopra-sa.com/>.
- [103] S. Nunomura, A. Minowa, H. Sai, M. Kondo, *Mie scattering enhanced near-infrared light response of thin-film silicon solar cells*, Appl. Phys. Lett., vol. 97, 063507, 2010.
- [104] S.-C. Kong, J. J. Simpson, V. Backman, *ADE-FDTD Scattered-Field Formulation for Dispersive Materials*, IEEE Micro. Wire. Comp. Lett., vol. 18, pp. 4-6, 2008.
- [105] O. E. Daif, E. Drouard, G. Gomard, A. Kaminski, A. Fave, M. Lemiti, S. Ahn, S. Kim, P. R. Cabarrocas, H. Jeon, C. Seassal, *Absorbing one-dimensional planar photonic crystal for amorphous silicon solar cell*, Opt. Soc. A., Vol. 18, No. S3, A293, 2010.
- [106] D. Duche, P. Torchio, L. Escoubas, F. Monestier, J. J. Simon, F. Flory, G. Mathian, *Improving light absorption in organic solar cells by plasmonic contribution*, Sol. Energy Mater. Sol. Cells 93, pp. 1377-1382, 2009.
- [107] V. E. Ferry, L. A. Sweatlock, D. Pacifici, H. A. Atwater, *Plasmonic Nanostructure Design for Efficient Light Coupling into Solar Cells*, Nano Lett., vol. 8, no. 2, pp. 4391-4397, 2008.

- [108] L. Hu, G. Chen, *Analysis of Optical Absorption in Silicon Nanowire Arrays for Photovoltaic Applications*, Nano Lett., vol. 7, no. 11, pp. 3249-3252, 2007.
- [109] C. Lin, M. L. Povinelli, *Optical absorption enhancement in silicon nanowire arrays with a large lattice constant for photovoltaic applications*, Opt. Soc. A., vol. 17, no. 22, 19372, 2009.
- [110] J. Grandidier, M. G. Deceglie, D. M. Callahan, H. A. Atwater, *Simulation of solar cell absorption enhancement using resonant modes of a nanosphere array*, J. Photon. Energy 2, 024502, 2012.
- [111] C. C. Chao, C. M. Wang, J. Y. Chang, *Spatial distribution of absorption in plasmonic thin film solar cells*, Opt. Soc. A., vol. 18, no. 11, 11763, 2010.
- [112] R. Dewan, M. Marinkovic, R. Noriega, S. Phadke, A. Salleo, D. Knipp, *Light trapping in thin-film silicon solar cells with submicron surface texture*, Opt. Soc. A., vol. 17, no. 25, 23058, 2009.
- [113] V. E. Ferry, A. Polman, H. A. Atwater, *Modeling Light Trapping in Nanostructured Solar Cells*, ACS Nano, vol. 5, no. 12, pp. 10055-10064, 2011.
- [114] J. R. Nagel, M. A. Scarpulla, *Enhancement absorption in optically tin solar cells by scattering from embedded dielectric nanoparticles*, Opt. Soc. A., vol. 18, no. S2, A139, 2010.
- [115] W. Wang, S. Wu, K. Reinhardt, Y. Lu, S. Chen, *Broadband Light Absorption Enhancement in Thin-Film Silicon Solar Cells*, Nano Lett., vol. 10, pp. 2012-2018, 2010.
- [116] S. Ramo & J. R. Whinnery & T. V. Duzer, *Fields and Waves in Communication Electronics*, Third Edition. John Wiley Sons, 1993.
- [117] InsideOptics Co., Ltd., Raywiz-FDTD, <http://www.insideoptics.com>.
- [118] A. Williams, S. Barrus, R. K. Morley, P. Shirley, *An efficient and robust ray-box intersection algorithm*, J. Graph. Tools 10, pp. 49-54, 2005.
- [119] C. J. Brinker, M. S. Harrington, *Sol-Gel derived antireflective coatings for silicon*, Sol. Energy Mater., Vol. 5, pp. 159-172, 1981.



- [120] F. W. Sexton, *Plasma Nitride AR coatings for silicon solar cells*, Sol. Energy Mater., Vol. 7, pp. 1-14, 1982.
- [121] F. Rubio, J. Denis, J. M. Albella, J. M. Martinez-Duart, *Sputtered Ta<sub>2</sub>O<sub>5</sub> antireflection coating for silicon solar cells*, Thin Solid Films, Vol. 90, pp. 405-408, 1982.
- [122] R. B. Pettit, C. J. Brinker, C. S. Ashley, *Sol-Gel double-layer antireflection coating for silicon solar cells*, Solar Cells, Vol. 15, pp. 267-278, 1985.
- [123] G. A. Landis, P. P. Jenkins, US Patent 5,261,970, 1993.
- [124] J. Müller, B. Rech, J. Springer, M. Vanecek, *TCO and light trapping in silicon thin film solar cells*, Solar Energy, Vol. 77, pp. 917-930, 2004.
- [125] B. Engquist, A. Majda, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comput., Vol. 31, pp. 629-651, 1977.
- [126] G. Mur, *Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic field equations*, IEEE Trans. Electrom. Compat., Vol. 23, pp. 377-382, 1981.
- [127] J. P. Bérenger, *A Perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., Vol. 114, pp. 185-200, 1994.
- [128] J. P. Bérenger, *Perfectly Matched Layer (PML) for Computational Electromagnetics*, Morgan & Claypool, 2007.
- [129] T. B. Yu, B. h. Zhou, B. Chen, *An unsplit formulation of the Berenger's PML absorbing boundary condition for FDTD meshes*, IEEE Microw. Wirel. Comp. Lett., Vol. 13, pp. 348-350, 2003.
- [130] Z. S. Sacks, D. M. Kingsland, R. Lee, J. F. Lee, *A perfectly matched anisotropic absorber for use as an absorbing boundary condition*, IEEE Trans. Antennas Propagat., Vol. 43, pp. 1460-1463, 1995.
- [131] J. A. Roden, S. D. Gedney, *Convolutional PML (CPML): An efficient FDTD implementation of the CFS-PML for arbitrary media*, Microw. Opt. Technol. Lett., Vol. 27, pp. 334-339, 2000.

# Appendix I

Optical properties of Crystalline Silicon from 300 - 900 nanometer wavelength.

Lambda(nm)	n	k
300	5.0049912	4.1596685
310	5.0101858	3.5847390
320	5.0236846	3.2999256
330	5.0982427	3.1236195
340	5.2216784	3.0108910
350	5.4345533	2.9883457
360	6.0488039	2.9919859
370	6.8181703	2.0307209
380	6.4700506	0.9656266
390	5.9425816	0.5581326
400	5.5964000	0.2840600
410	5.3217300	0.2038500
420	5.1077500	0.1489400
430	4.9325300	0.1128400
440	4.7871800	0.0902800
450	4.6666400	0.0766100
460	4.5651300	0.0678800
470	4.4788000	0.0619100
480	4.4046100	0.0574100
490	4.3401400	0.0536600
500	4.2830500	0.0503300
510	4.2326300	0.0472800
520	4.1872200	0.0444100
530	4.1461900	0.0417100
540	4.1089500	0.0391500
550	4.0750000	0.0367400
560	4.0438200	0.0344500
570	4.0150900	0.0322800
575	4.0015600	0.0312500
580	3.9884500	0.0302400
590	3.9638300	0.0283100

600	3.9410100	0.0264900
610	3.9196700	0.0247700
620	3.8996800	0.0231500
630	3.8809500	0.0216300
640	3.8633800	0.0201900
650	3.8469300	0.0188300
660	3.8313800	0.0175500
670	3.8166600	0.0163600
680	3.8027500	0.0152300
690	3.7896700	0.0141600
700	3.7772300	0.0131700
710	3.7654100	0.0122300
720	3.7541800	0.0113400
730	3.7434800	0.0105100
735	3.7384000	0.0101200
740	3.7333800	0.0097300
750	3.7237400	0.0090000
760	3.7145300	0.0083200
770	3.7056600	0.0076700
780	3.6972900	0.0070700
790	3.6892500	0.0065000
800	3.6815400	0.0059700
810	3.6741800	0.0054800
820	3.6671100	0.0050100
830	3.6603100	0.0045800
840	3.6537700	0.0041800
850	3.6474600	0.0038000
860	3.6414200	0.0034500
870	3.6357100	0.0031300
880	3.6301400	0.0028200
890	3.6247300	0.0025400
900	3.6195100	0.0022800

## 국문초록

본 논문에서는 무작위의 복잡한 3차원 표면 형상을 가진 물체를 평균적으로  $O(\log N)$  시간 복잡도를 가지고 교차검사를 수행할 수 있는 새로운 알고리즘을 기반으로 박막형 태양전지 (Thin Film Solar Cell)의 흡수 효율을 효과적으로 시뮬레이션 하기 위한 방법을 논하였다.

3차원 피라미드 형상을 박막형 태양전지 표면에 양각한 경우, 3차원 표면 형상의 크기와 밀도가 흡수 효율에 영향을 미치게 되므로 이들에 대한 최적 설계 값들을 찾고자 시뮬레이션 방법을 이용한다.

본 연구에서는 무수히 많은 3차원 표면 형상들에 대한 최적의 교차검사 알고리즘을 kd-tree 가속화 구조를 변형하여 개발하였고, 이를 광선 추적 법 (Ray tracing)에 적용하여 평균 교차검사 시간을  $O(\log N)$ 으로 하는 새로운 광선 추적에 의한 시뮬레이션 방법을 고안하였다. 이 방법은 기존 연구들에서 한번도 연구되지 않은 새로운 방법으로, 표면 형상들을 실제 객체로 인지하여 교차검사를 수행하면서도 종래의  $O(N) \sim O(N^2)$  교차검사 시간을  $O(\log N)$ 으로 단축시키는 결과를 얻었다. 또한 이전 연구들에서 반사율 시뮬레이션에 의존하여 간접적으로 계산하던 에너지 흡수 효율을 직접적으로 시뮬레이션 함은 물론이고 각 층별 에너지 흡수율을 간섭현상을 반영하여 직접적으로 시뮬레이션 할 수 있는 방법을 고안하였다. 이 알고리즘의 효율성 및 정확성은 다른 알고리즘과의 수행 시간 비교 및 실측 자료와 시뮬레이션 결과와의 오차 분석을 통해 검증 하였다.

박막형 태양전지의 크기가 나노미터 단위로 작아진 경우, 유한차분 시간영역 (FDTD)법을 이용하게 되는데, 현재까지 연구된 방법들로는 적은 전산 자원을 사용하여 빠른 시간 내에 정확한 흡수 에너지를 계산하는데 한계가 있었다. 본 연구에서는 이 문제를 해결하기 위하여

자유로운 형태의 시스템에 대해서 각 물질들의 연속된 경계 면을 효율적으로 추출하여 Poynting 이론의 발산 (Divergence) 부분의 식을 적용하는 방법으로 적은 전산 자원으로 빠른 시간 내에 상당한 정확도를 가지고 흡수 에너지를 계산할 수 있는 방법을 고안하였다. 이 방법의 정확성은 해석 가능한 모델의 Mie 확산 모델의 계산 결과와 시뮬레이션 결과를 비교함으로써 검증하였으며, 곡면 모델에 대해서는 곡률 반경을 변화시켜가며 시뮬레이션 한 결과가 물리적 유의성을 나타냄을 보임으로써 검증 하였다.

주요어 : 박막형 태양전지, 광선 추적 법, 유한차분 시간영역 법, 교차검사, 흡수 에너지

학번 : 2010-20407

## 감사의 글

먼저 박사학위과정을 무사히 마치고 학위를 받을 수 있도록 지혜와 건강을 허락하신 하나님께 감사 드립니다.

적지 않은 나이의 학생임에도 불구하고 제자로 받아주시고 학위 과정 동안 세심한 배려로 학자의 자세와 자질에 대하여 지도하여 주시며, 또한 몸소 그 본을 보여주신 신동우 교수님께 감사 드립니다.

또한, 바쁘신 와중에도 시간을 허락하셔서 학위 논문을 검토하여 주시고 미흡한 부분을 지도하여 주신 정현교 교수님, 고흥석 교수님, 박춘재 교수님, 그리고 김임범 박사님께 감사 드립니다.

또한, 학위 과정 중 연구를 수행하고 연구 결과물들을 출판함에 있어 연관된 전문 분야의 지식을 공유하고 결과물에 대한 논의를 통해 연구 성과를 발전시킬 수 있도록 함께 연구해준 동생과 KIST 전자재료 연구실의 김원목 박사님, 이택성 박사님께 감사 드립니다.

그리고 학위 과정 동안 많은 나이 차이에도 불구하고 편안한 마음으로 연구실에 거할 수 있도록 배려해준 남현 박사, 이선정 박사, 룡택, 형석, 효은, 시환, 용진, 광신, 지운에게도 감사 드립니다.

또한, 항상 옆에서 기도와 사랑으로 지켜주신 할머니와 부모님께 감사 드립니다.

마지막으로 사랑하는 아내 영진 그리고 사랑하는 아들들 형조, 형민, 형준에게도 많은 시간을 함께 하지 못한 미안함과 사랑의 마음을 전합니다.